

Frontiers  
in  
Artificial  
Intelligence  
and  
Applications

# DATABASES AND INFORMATION SYSTEMS IV

Selected Papers from the Seventh  
International Baltic Conference DB&IS'2006

Edited by  
Olegas Vasilecas  
Johann Eder  
Albertas Caplinskas

**IOS**  
Press

VISIT...

LANZAROTE  
*Caliente*.COM

## DATABASES AND INFORMATION SYSTEMS IV

# Frontiers in Artificial Intelligence and Applications

FAIA covers all aspects of theoretical and applied artificial intelligence research in the form of monographs, doctoral dissertations, textbooks, handbooks and proceedings volumes. The FAIA series contains several sub-series, including “Information Modelling and Knowledge Bases” and “Knowledge-Based Intelligent Engineering Systems”. It also includes the biennial ECAI, the European Conference on Artificial Intelligence, proceedings volumes, and other ECCAI – the European Coordinating Committee on Artificial Intelligence – sponsored publications. An editorial panel of internationally well-known scholars is appointed to provide a high quality selection.

Series Editors:

J. Breuker, R. Dieng-Kuntz, N. Guarino, J.N. Kok, J. Liu, R. López de Mántaras,  
R. Mizoguchi, M. Musen and N. Zhong

## Volume 155

*Recently published in this series*

- Vol. 154. M. Duží et al. (Eds.), Information Modelling and Knowledge Bases XVIII
- Vol. 153. Y. Vogiazou, Design for Emergence – Collaborative Social Play with Online and Location-Based Media
- Vol. 152. T.M. van Engers (Ed.), Legal Knowledge and Information Systems – JURIX 2006: The Nineteenth Annual Conference
- Vol. 151. R. Mizoguchi et al. (Eds.), Learning by Effective Utilization of Technologies: Facilitating Intercultural Understanding
- Vol. 150. B. Bennett and C. Fellbaum (Eds.), Formal Ontology in Information Systems – Proceedings of the Fourth International Conference (FOIS 2006)
- Vol. 149. X.F. Zha and R.J. Howlett (Eds.), Integrated Intelligent Systems for Engineering Design
- Vol. 148. K. Kersting, An Inductive Logic Programming Approach to Statistical Relational Learning
- Vol. 147. H. Fujita and M. Mejri (Eds.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the fifth SoMeT\_06
- Vol. 146. M. Polit et al. (Eds.), Artificial Intelligence Research and Development
- Vol. 145. A.J. Knobbe, Multi-Relational Data Mining
- Vol. 144. P.E. Dunne and T.J.M. Bench-Capon (Eds.), Computational Models of Argument – Proceedings of COMMA 2006
- Vol. 143. P. Ghodous et al. (Eds.), Leading the Web in Concurrent Engineering – Next Generation Concurrent Engineering

ISSN 0922-6389

# Databases and Information Systems IV

Selected Papers from the Seventh International Baltic Conference  
DB&IS'2006

Edited by

**Olegas Vasilecas**

*Vilnius Gediminas Technical University, Lithuania*

**Johann Eder**

*University of Vienna, Austria*

and

**Albertas Caplinskas**

*Institute of Mathematics and Informatics, Lithuania*

**IOS**  
Press

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

© 2007 The authors and IOS Press.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without prior written permission from the publisher.

ISBN 978-1-58603-715-4

Library of Congress Control Number: 2007920254

*Publisher*

IOS Press

Nieuwe Hemweg 6B

1013 BG Amsterdam

Netherlands

fax: +31 20 687 0019

e-mail: [order@iospress.nl](mailto:order@iospress.nl)

*Distributor in the UK and Ireland*

Gazelle Books Services Ltd.

White Cross Mills

Hightown

Lancaster LA1 4XS

United Kingdom

fax: +44 1524 63232

e-mail: [sales@gazellebooks.co.uk](mailto:sales@gazellebooks.co.uk)

*Distributor in the USA and Canada*

IOS Press, Inc.

4502 Rachael Manor Drive

Fairfax, VA 22032

USA

fax: +1 703 323 3668

e-mail: [iosbooks@iospress.com](mailto:iosbooks@iospress.com)

LEGAL NOTICE

The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

# Conference Committee

## Advisory Committee

Janis Bubenko, Royal Institute of Technology, Sweden

Arne Solvberg, Norwegian University of Science and Technology, Norway

## Programme Co-Chairs

Albertas Caplinskas, Institute of Mathematics and Informatics, Lithuania

Johann Eder, University of Vienna, Austria

Olegas Vasilecas, Vilnius Gediminas Technical University, Lithuania

## Programme Committee

Witold Abramowicz, Poland

Viktor Aleksejev, Byelorussia

Majed Al-Mashari, Saudi Arabia

Marko Bajec, Slovenia

Janis Barzdins, Latvia

Maria Bielikova, Slovakia

Juris Borzovs, Latvia

Sjaak Brinkkemper, The Netherlands

Bostjan Brunen, Slovenia

Janis A. Bubenko, Sweden

Rimas Butleris, Lithuania

Sharma Chakravarthy, USA

Lawrence Chung, USA

Cesar Alberto Collazos, Colombia

Jose Cordeiro, Portugal

Heitor Augustus Xavier Costa, Brazil

Bogdan Czejdo, USA

Oguz Dikenelli, Turkey

Dale Dzemydiene, Lithuania

Hans-Dieter Ehrich, Germany

Andre Flory, France

Maria Grazia Fugini, Italy

Franca Garzotto, Italy

Janis Grundspenkis, Latvia

Giancarlo Guizzardi, Italy

Hele-Mai Haav, Estonia

Juhani Iivari, Finland

Mirjana Ivanovic, Montenegro

Leonid Kalinichenko, Russia

Ahto Kalja, Estonia

Audris Kalnins, Latvia

Gunter Saake, Germany

Simonas Saltenis, Denmark

Marite Kirikova, Latvia

Gábor Knapp, Hungary

Christian Koncilia, Austria

Manolis Koubarakis, Greece

John Krogstie, Norway

Rein Kuusik, Estonia

Sergej Kuznetsov, Russia

Patrick Lambrix, Sweden

Michael Lang, Ireland

Pericles Loucopoulos, UK

Audrone Lupeikiene, Lithuania

Kalle Lyytinen, USA

Leszek Maciaszek, Australia

Yannis Manolopoulos, Greece

Rainer Manthey, Germany

Saulius Maskeliunas, Lithuania

Mihhail Matskin, Sweden

Kai Mertins, Germany

Audris Mockus, USA

Lina Nemuraite, Lithuania

Jorgen Fischer Nilsson, Denmark

Kjetil Norvag, Norway

Boris Novikov, Russia

Algirdas Pakstas, UK

Oskar Pastor, Spain

Jaan Penjam, Estonia

Klaus Pohl, Germany

Jaroslav Pokorny, Czech Republic

Henrikas Pranevicius, Lithuania

Boris Rachev, Bulgaria

Daniela Rosca, USA

Ernest Teniente, Spain

Bernhard Thalheim, Germany

Klaus Dieter Schewe, New Zealand  
 Joachim Schmidt, Germany  
 Timothy K. Shih, Taiwan  
 Rimvydas Simutis, Lithuania  
 Guttorm Sindre, Norway  
 Eva Söderström, Sweden  
 Arne Solvberg, Norway  
 William Song, UK  
 Janis Stirna, Sweden  
 Julius Stuller, Czech Republic

Enn Tyugu, Estonia  
 Irma Valatkaite, Lithuania  
 Gottfried Vossen, Germany  
 Benkt Wangler, Sweden  
 Robert Wrembel, Poland  
 Mudasser Wyne, USA  
 Tatyana Yakhno, Turkey  
 Arkady Zaslavsky, Australia  
 Roberto Zicari, Germany

### **Additional Referees**

Raman Adaikkalavan, USA  
 Rose-Mharie Ahlfeldt, Sweden  
 Andrzej Bassara, Poland  
 Sebastian Bossung, Germany  
 Xiaofeng Du, United Kingdom  
 Silke Eckstein, Germany  
 Hugo Estrada, Mexico  
 Miguel Garcia, Germany  
 Joe Geldart, United Kingdom  
 Martin Henkel, Sweden  
 Jesper Holgersson, Sweden  
 Maik Kollmann, Germany  
 Andreas Kupfer, Germany  
 Dejan Lavbič, Slovenia  
 Marek Lehmann, Austria  
 Kira Vyatkina, Russia

Marion Lepmets, Estonia  
 Carolin Letz, Germany  
 Alexandros Nanopoulos, Greece  
 Apostolos Papadopoulos, Greece  
 Bronius Paradauskas, Lithuania  
 Aida Pliuskeviciene, Lithuania  
 Regimantas Pliuskevicius, Lith.  
 Ingo Schmitt, Germany  
 Joachim Schwierén, Germany  
 Hans-Werner Sehring, Germany  
 Mattias Strand, Sweden  
 Aditya Telang, USA  
 Roland Turba, Germany  
 Karol Wieloch, Poland  
 Damjan Vavpotič, Slovenia

### **Organising Chair**

Olegas Vasilecas, Information Systems Research Laboratory, Vilnius Gediminas Technical University, Lithuania

### **Organising Committee**

Algirdas Ciucelis, Vilnius Gediminas Technical University, Lithuania  
 Juris Borzovs, Institute of Mathematics and Computer Science, Latvia  
 Ahto Kalja, Institute of Cybernetics at TTU, Estonia  
 Virgilijus Sakalauskas, Vilnius University Kaunas Faculty of Humanities, Lithuania  
 Algirdas Pakstas, London Metropolitan University, UK  
 Adomas Svirskas, Vilnius University, Lithuania  
 Diana Bugaite, Vilnius Gediminas Technical University, Lithuania  
 Dale Dzemydiene, Institute of Mathematics and Informatics, Lithuania  
 Dalia Kriksciuniene, Vilnius University Kaunas Faculty of Humanities, Lithuania  
 Audrone Lupeikiene, Institute of Mathematics and Informatics, Lithuania  
 Saulius Maskeliunas, Institute of Mathematics and Informatics, Lithuania  
 Arunas Ribikauskas, Vilnius Gediminas Technical University, Lithuania  
 Rimvydas Simutis, Vilnius University Kaunas Faculty of Humanities, Lithuania  
 Sergejus Sosunovas, Vilnius Gediminas Technical University, Lithuania



## Preface

This volume contains the best papers presented at the 7th International Baltic Conference on Databases and Information Systems (BalticDB&IS'2006). The series of Baltic DB&IS conferences has been initiated by Janis Bubenko jr. and Arne Solvberg in 1994. The conferences are highly international and bring together academics and practitioners from the entire world. They are organized by the Baltic countries in turn. The first conference was held in Trakai (1994), then followed conferences in Tallinn (1996), Riga (1998), Vilnius (2000), Tallinn (2002), Riga (2004), and again in Vilnius. The conference BalticDB&IS'2006 took place on July 3–6, 2006. It was organized by the Department of Information Systems (Vilnius Gediminas Technical University) and Software Engineering Department (Institute of Mathematics and Informatics). The conference has been approved by the IEEE Communication Society for Technical Co-sponsorship.

The call for papers attracted 84 submissions from 21 countries. In a rigorous reviewing process the international program committee selected 48 papers for the presentation at the Conference and 27 for publishing in the Proceedings published by IEEE. After the Conference the program committee selected 20 best papers to be published in this volume. All these papers have been extended significantly and rewritten completely. They have been reviewed by at least 3 reviewers from different countries who evaluated their originality, significance, relevance, and presentation and found their quality suitable for the publication in this volume. These papers present original results in business modeling and enterprise engineering, database research, data engineering, data quality and data analysis, IS engineering, Web engineering, and application of AI methods. We hope that the presented results will contribute to the further development of research in DB and IS field.

We would like to express our warmest thanks and acknowledgements to all the people who contributed to BalticDB&IS'2006:

- the authors, who submitted papers to the conference,
- the members of the international program committee and the additional referees, who voluntarily reviewed the submitted papers in order to ensure the quality of the scientific program,
- the sponsors of the conference, who made this conference possible,
- all the local organizing team voluntarily giving their time and expertise to ensure the success of the conference.

We express our special thanks to Audrone Lupeikiene for all his assistance during the preparation of this volume.

Olegas Vasilecas  
Johann Eder  
Albertas Caplinskaskas

This page intentionally left blank

# Contents

Conference Committee	v
Preface	vii
<i>Olegas Vasilecas, Johann Eder and Albertas Caplinskas</i>	
<b>Invited Papers</b>	
Events and Rules for Java: Using a Seamless and Dynamic Approach	3
<i>Sharma Chakravarthy, Rajesh Dasari, Sridhar Varakala and Raman Adaikkalavan</i>	
On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models	18
<i>Giancarlo Guizzardi</i>	
Pattern Repositories for Software Engineering Education	40
<i>Hans-Werner Sehring, Sebastian Bossung, Patrick Hupe, Michael Skusa and Joachim W. Schmidt</i>	
<b>Business Modeling and Enterprise Engineering</b>	
A Technique for the Prediction of Deadline-Violations in Inter-Organizational Business Processes	57
<i>Johann Eder, Horst Pichler and Stefan Vielgut</i>	
Interactions at the Enterprise Knowledge Management Layer	72
<i>Saulius Gudas and Rasa Brundzaite</i>	
<b>Databases</b>	
Methods for a Synchronised Evolution of Databases and Associated Ontologies	89
<i>Andreas Kupfer, Silke Eckstein, Britta Störmann, Karl Neumann and Brigitte Mathiak</i>	
A Heuristic Approach to Fragmentation Incorporating Query Information	103
<i>Hui Ma, Klaus-Dieter Schewe and Markus Kirchberg</i>	
Introducing Softness into Inductive Queries on String Databases	117
<i>Ieva Mitasiunaite and Jean-François Boulicaut</i>	
<b>Data Engineering</b>	
A Multiple Correspondence Analysis to Organize Data Cubes	133
<i>Riadh Ben Messaoud, Omar Boussaid and Sabine Loudcher Rabaséda</i>	

Model-Driven Development for Enabling the Feedback from Warehouses and OLAP to Operational Systems	147
<i>Lina Nemuraite, Jurgita Tonkunaite and Bronius Paradauskas</i>	

## **Data Quality and Data Analysis**

A Time-Series Representation for Temporal Web Mining Using a Data Band Approach	161
<i>Mireille Samia and Stefan Conrad</i>	
The Framework for Business Rule Based Software Modeling: An Approach for Data Analysis Models Integration	175
<i>Olegas Vasilecas and Aidas Smaizys</i>	

## **IS Engineering**

Aspect-Oriented Use Cases and Crosscutting Interfaces for Reconfigurable Behavior Modeling	189
<i>Lina Nemuraite and Milda Balandyte</i>	
Integrated Enterprise Information Systems: Thinking in Component Concepts	203
<i>Audrone Lupeikiene</i>	

## **Web Engineering**

A Web Service-Oriented Architecture for Implementing Web Information Systems	219
<i>Flavius Frasincar, Geert-Jan Houben and Peter Barna</i>	
Adaptive Support of Inter-Domain Collaborative Protocols Using Web Services and Software Agents	233
<i>Adomas Svirskas, Michael Wilson, Bob Roberts and Ioannis Ignatiadis</i>	
Using a Rule Language for Capturing Semantics in Web-Based Systems	249
<i>Tanel Tammet, Hele-Mai Haav, Vello Kadarpiik and Marko Kääramees</i>	

## **Application of AI Methods**

A Concept Map Based Intelligent System for Adaptive Knowledge Assessment	263
<i>Alla Anohina and Janis Grundspenkis</i>	
Semantic Interoperability Between Functional Ontologies	277
<i>Nacima Mellal, Richard Dapoigny and Laurent Foulloy</i>	
Principles of Model Driven Architecture in Knowledge Modeling for the Task of Study Program Evaluation	291
<i>Oksana Nikiforova, Marite Kirikova and Natalya Pavlova</i>	

Author Index	305
--------------	-----

## Invited Papers

This page intentionally left blank

# Events and Rules for Java: Using a Seamless and Dynamic Approach

Sharma CHAKRAVARTHY<sup>a</sup>, Rajesh DASARI<sup>a</sup>, Sridhar VARAKALA<sup>a</sup> and  
Raman ADAIKKALAVAN<sup>b, 1</sup>

<sup>a</sup>ITLab & CSE Department, The University of Texas at Arlington

<sup>b</sup>CIS Department, Indiana University South Bend

**Abstract:** Proponents of active systems have proposed Event-Condition-Action (ECA) rules, a mechanism where behavior is invoked automatically as a response to events but without user or application intervention. The environment (the programming language and the operating system) in which a system is built influences how the event detector is designed and implemented. Sentinel provided active capability to an object-oriented database environment implemented in C++. However, C++ environment had certain limitations that proved deterrent to implementing some of the features of active capability. This paper discusses the re-designing and implementation of the active subsystem in the Java environment. Main motivations behind our objective of re-designing and implementing the active subsystem in the Java environment include: i) to overcome the limitations of the C++ environment, and ii) to exploit some of the capabilities provided by the Java environment that are critical for an active system. It also provides a novel approach for supporting the creation of rules, and composite and temporal events *dynamically* at run time, which is inevitable for several classes of monitoring applications. This avoids recompilation and restart of the system which are inappropriate in many environments that require fine-tuning of rules on the fly. It provides a generic set of classes that are designed to handle rules dynamically. This set of generic classes is application-independent making the system a general-purpose tool.

**Keywords:** ECA rules for Java, event detection graphs, dynamic events and rules

## Introduction

Situations can be monitored by defining Event-Condition-Action rules on the events that are of interest in the system. The imminent need for security in the real world has led to the development of various applications ranging from Coast Patrolling mechanisms to Network Management protocols. There is a need to continually monitor some applications 24x7 which may be difficult if a user is in the loop. As an example, radar personnel in a military environment have to constantly monitor air traffic. Alternately, the ECA paradigm can be applied to monitor the radar and respond to events without user intervention. An ECA rule consists of three components – an event, a condition, and an action. According to the ECA rule semantics, whenever an event occurs, a condition is checked and the action is carried out if the condition evaluates to true. The behavior exhibited by applications by means of ECA rules, (i.e., an action

---

<sup>1</sup> Corresponding Author: E-mail: sharma@cse.uta.edu

being carried out as a consequence of a certain event) is known as the active behavior. Programming languages, database systems and GUIs are being enhanced to provide explicit support for active behavior due to the large range of applications that naturally express their semantics using this paradigm. Active behavior is also useful to those applications that require monitoring situations and reacting to them without user or application intervention. The process of incorporating active behavior entails the following steps: i) defining the event and the rules associated with the event; ii) detecting the event when it occurs; and iii) reacting to the event, i.e., executing rules and carrying out the operations specified in the actions (rules) defined over the event. As described in Anwar et al. [1], event detection is considerably complex for an object-oriented environment and furthermore, compile and runtime issues need to be addressed. Sentinel supports an expressive event specification language (termed Snoop [2]), a Local Event Detector that detects events and a Rule Scheduler that schedules the rules and executes them. Owing to the C++ implementation of OpenOODB (from Texas Instruments), all the components of Sentinel were developed in C++. As a result, it had a bearing on the active behavior supported, like the types of event parameters allowed, how the condition and action parts of the rule were modeled.

The main aim of this paper is to design and develop a system that provides support for events and rules in Java applications (whether it is a DBMS developed in Java or a general Java application) using a *seamless* approach. This paper also discusses how events and rules can be created and maintained *dynamically*. This paper addresses the issues involved in the redesign of the subsystem that provides active capability while moving from the C++ environment to the Java environment, as well as overcoming the limitations of the C++ environment. Although both C++ and Java are object-oriented programming languages, there are some fundamental differences in ideologies and capabilities provided by them. For example, Java provides mechanisms to obtain some dynamic information about the application objects during runtime whereas C++ does not provide any such mechanism.

Outline: Section 1 summarizes the ECA rule semantics and event operators. Section 2 describes related work. Section 3 explains the design issues, approach taken and its rationale. Section 4 describes the implementation details and Section 5 has conclusions.

## 1. ECA Rule Semantics

This section summarizes various Snoop event operators and rules and their associated semantics described in [2, 3].

*“An event is an instantaneous and atomic (happening completely or not at all) occurrence”*. For example, in the relational database domain, database operations such as retrieve, insert, update, and delete can be defined as events. An event  $E$  is a function from the time domain onto the Boolean values, True and False. The function is given by

$$E(t) = \begin{array}{ll} \text{True} & \text{if an event of type } E \text{ occurs at time point } t \\ \text{False} & \text{otherwise} \end{array}$$

A *primitive event* is an event expression comprising a single event in the domain of consideration. A *composite event* is defined recursively, as an event expression using set of primitive event expressions, event operators, and composite event expressions constructed up to that point. Furthermore, composite events can be detected in different



event consumption modes (a.k.a. parameter contexts). On the other hand, semantics of a *primitive event* is identical in all contexts. *Rules* can be defined on both primitive and composite events. Whenever an event (primitive or composite) is detected, the rules associated with that event are executed based on their priority and coupling mode.

Some of the Snoop event operators [2, 3] are as described below. “E” is used to represent an event type and “e” is used to represent an instance “E”. Superscripts are used to denote the relative time of occurrence with respect to events of the same type.

- **OR (V):** Disjunction of two events  $E_1$  and  $E_2$  i.e.,  $E_1 \vee E_2$ , occurs when either one occurs.
- **AND (A):** Conjunction of two events  $E_1$  and  $E_2$ , denoted by  $E_1 \wedge E_2$  occurs when both  $E_1$  and  $E_2$  occur, irrespective of their order of occurrence.
- **SEQUENCE (;):** Sequence of two events  $E_1$  and  $E_2$ , denoted by  $E_1;E_2$  occurs when  $E_2$  occurs provided  $E_1$  has already occurred.
- **NOT (–):** The *not* operator, denoted by  $\neg(E_2)[E_1,E_3]$  detects the non-occurrence of the event  $E_2$  in the closed interval formed by  $E_1$  and  $E_3$ .
- **Aperiodic (A, A\*):** The Aperiodic operator A ( $E_1, E_2, E_3$ ) allows one to express the occurrence of an aperiodic event  $E_2$ . The event A is signaled each time  $E_2$  occurs during the half-open interval defined by  $E_1$  and  $E_3$ . A can occur zero or more times. Similarly, cumulative version of Aperiodic operator  $A^*(E_1,E_2,E_3)$  occurs only once when  $E_3$  occurs as it accumulates the occurrences of  $E_2$  in the half-open interval formed by  $E_1$  and  $E_3$ .
- **Periodic (P, P\*):** A periodic event is an event E that repeats itself with a constant and finite amount of time. Only a time specification is meaningful for E. The notation used for expressing a periodic event is P ( $E_1, [t], E_3$ ) where  $E_1$  and  $E_3$  are events and  $[t]$  (or E) is the (positive) time specification. P occurs for every  $[t]$  in the half-open interval ( $E_1,E_3$ ). Similarly, P\* is the cumulative version of the P operator.

**Parameter Contexts:** Parameter contexts capture the application semantics while computing the parameters of composite events that are not unique. They disambiguate the parameter computation and at the same time accommodate a wide range of application requirements [4]. The contexts are defined using the notion of initiator and terminator events. An initiator event initiates or starts the detection of a composite event and a terminator event completes the detection of the composite event. For example, for the sequence event  $E_1;E_2$ ,  $E_1$  will be the initiator event and  $E_2$  will be the terminator event. We will not discuss the contexts further, as the focus of this paper is design, architecture and implementation of an event detector.

**Coupling Modes for Java Applications:** Coupling modes specify when a rule should be executed relative to the event that is firing the rule. They were initially proposed for a transaction based execution environment such as a DBMS. But, the execution environment of object-oriented applications, the environment for which active capability is being proposed in this paper, is not transaction-based. In the definitions below, the difference in the meaning of the coupling modes between a transaction based environment and a non-transaction based environment (the current environment) is also explained. i) *Immediate:* In this, the fired rule is executed immediately after the event is detected, in which case the execution of the triggering transaction is suspended until the rule is executed. In a non-transaction-based environment, the thread of execution of the application or the rule that raises the event is suspended until the rule is executed. ii) *Deferred:* In this, the execution of a fired rule is deferred to the end of the transaction. However, there are no transaction boundaries

in a non-transaction-based environment. For this purpose, we define two events named `executeDeferredRules` and `discardDeferredRules`. All the deferred rules are accumulated from the beginning of the application and executed when the former event is explicitly raised by the application. The application can also raise the latter event at which point all the deferred rules accumulated thus far are discarded. It is implicit that whenever a deferred rule is triggered, it is accumulated. The accumulated rules are either executed or discarded depending on one of the above events explicitly raised by the application.

## 2. Related Work

We will describe some of the newer event-based systems in this section. We will not elaborate on some of the earlier work, such as Ode [5, 6], ADAM [7], and Samos [8, 9], that are similar to Sentinel [3, 10, 11] in some ways and differ in some other ways.

**Vitria BusinessWare Process Automator** [12]: It provides a modeling environment that captures business objects, events, rules and processes and uses them to build a collection of business policies. It incorporates four modeling techniques into an integrated process-development environment: business object models, business event models, business process (or “state”) models and business rules. It uses a graphical modeling language to create process charts that describe the different stages of a business process. Business rules and policies are expressed in ECA sequences. A rule condition compares a process *variable* with some value and such relational comparisons can be connected through logical operators. The rule action invokes methods on process objects. It has no support for composite events and context based event detection unlike our system.

**WebLogic Events** [13]: It is an event notification and management service. It provides event registration and notification between applications across a network. It has a server that stores the event registrations from any application across the network. Event registrations are stored in a Topic Tree, which is a hierarchical, n-ary tree of period-separated words, where each word represents a node at a particular level in the tree. The whole event service flows through the Topic Tree. Since a registration is associated with a single evaluate and action pair, only a single rule can be defined on an event whereas our system allows for multiple rules to be defined on an event.

## 3. Design of the Event Detector

The event detector should provide APIs to user applications for defining primitive and composite events and to define rules on the events. It should also contain the detection logic for detecting composite events in any of the contexts. Finally, when events are detected, the rules defined on those events should be executed based on their coupling mode and priority.

### 3.1. Types of Events

A Java application is a collection of classes and each class is a collection of attributes and methods. The application logic mostly consists of invoking methods on objects of

these classes. For this reason, method invocations are treated as primitive events. An event occurs either at the beginning or at the end of the method. When a method of a class is defined as a primitive event (a.k.a. *class level* event), an event occurs when any instance of the class invokes the method. On the other hand, if it is defined as an *instance level* event, the event is detected only when a particular instance invokes the method. The instance is specified in the definition of the instance level event. Class level events can be specified within the class definition. Instance level events can be specified only where the instance is declared.

#### **EVENT begin (setPriceBegin) void setPrice(float price)**

Event shown above is a class level primitive event named ‘setPriceBegin’ specified using Snoop syntax. It occurs at the beginning of the ‘setPrice’ method. Syntax includes name of the event, method signature and event modifier (begin or end). All primitive events should be named, since they can be used in a composite event expression.

#### **EVENT end (setPriceIBMEnd:IBM) void setPrice(float price)**

Event shown above is an instance level primitive event. IBM is the name of the instance of the class in which the ‘setPrice’ method is defined. This event occurs only when the IBM instance invokes the ‘setPrice’ method. Instance should be predefined.

Below shown is a complex composite event defined using Snoop operators.

#### **EVENT eSeqOrAnd = (e1 ; e2) | (e3 $\wedge$ e4)**

Temporal events that are detected at a specific clock tick can also be defined. There are two types of temporal events – *absolute* and *relative*, where the former is specified to occur at a point in time and the latter is defined by specifying a relative time expression in the definition of a composite event using one of P, P\* or PLUS operators. Absolute temporal event **eAbs** defined below occurs at November 15, 2006 at 10:40:40. Relative temporal event **ePeriodic** occurs every 5 minutes 10 seconds once initiated by **setPriceIBMEnd** and is terminated whenever event **buyStockIBMBegin** occur.

#### **EVENT eAbs = [10:40:40/11/15/2006]**

#### **EVENT ePeriodic = P(setPriceIBMEnd, [ 5 min 10 sec], buyStockIBMBegin)**

### *3.2. Event Parameters*

Events are associated with a set of parameters that are passed to the condition and action portions of the rule. A primitive event is associated with a single set of parameters whereas a composite event is associated with sets of parameters, that is, one parameter set for each of its constituent primitive event. The rule depends upon the values of these parameters to take appropriate actions. Typically, a rule condition checks the values of one or more parameters of the event. The action part of the rule is executed only if the conditions return true. As events are defined as method invocations, formal arguments of those methods and the object that invoked the method are treated as the parameters for that event. Primitive data types (int, etc.), object data types defined by Java (String, etc.) and user defined data types can all be passed as arguments to a method. There should be a mechanism for collecting the parameters of an event, storing them, and then retrieving individual parameters from the parameter set. In Java, there is a generic Object data type that can contain a reference to an instance of any class type. This data type is used to store all the parameters in the parameter list.

Primitive data types are stored in the parameter list after converting them to their object equivalents. During parameter retrieval, the primitive value stored in the object is returned. Class data types are stored as generic objects and they are returned as is during parameter retrieval. In C++, there is no such generic data type that can store both primitive data types and pointers. Hence only primitive data types were supported as event parameters in C++. The section on implementation describes the data structures used for storing and retrieving the parameters.

### 3.3. Rules

Rule condition and action are defined as methods associated with some class as all the execution in Java applications are through member methods. In order to execute methods specified as a condition or an action, one needs to get a reference to that object at run time. As Java supports references to class methods and hence conditions and actions can be implemented as methods in Java. If the condition and action are implemented as methods of a class, all the attributes of the class can be used in condition checking as well as in action execution. This is not possible if conditions and actions are implemented as functions, as in C++. As events and rules are defined at independent points of time, it is possible to define an event on one class and a rule associated with that event in another class. When an event is raised on the instance of one class, a default instance of another class is automatically created for executing condition and action methods by our system. It is also possible to specify instances that should be used to execute condition/action methods.

Java reflection is used to obtain the references to the condition and action methods defined in a class. In order to do that, the arguments passed to a condition and action method should be known to the event detector. Also, from a user's point of view, it is not meaningful to overload these methods. Hence, only a fixed number and type of arguments is allowed for these methods. Therefore, all condition and action methods take a single argument of type 'ListOfParameterLists'. This data type, defined in the event detector, stores a list of 'parameter lists'. A parameter list is a set of parameters associated with a primitive event.

If the user wants a composite event to be detected in two different contexts and defines rules on them, he/she has to specify the same event expression in two event definitions (one for each context) and define rules on each of them. This results in two event definitions although both of them contain the same event expression. This can be avoided if the context information is specified as part the rule definition, instead of specifying it as part of the event definition. Now, the composite event is defined only once and the rule defined in a particular context is fired only if the event is detected in the same context. In addition to condition, action, and context other attributes such as coupling mode and priority are also associated with a rule.

#### **RULE r1 [setPriceEnd, checkPrice, buyStock, RECENT, IMMEDIATE, NOW, 5]**

The above class-level rule r1 has 'checkPrice' and 'buyStock' methods as condition and action, respectively, immediate coupling mode with a priority of 5. This rule is fired only when event 'setPriceEnd' is detected in the *recent* context.

#### **RULE r2 [setPriceIBMEnd, checkPrice, buyStock]**

#### **RULE r3 [setPriceEnd, IBM, checkPrice, buyStock]**

**Instance Level Rules:** Rules defined on instance level events are called instance level rules. For example, rule **r2** is triggered when the instance level event ‘setPriceIBMEnd’ is detected. As will be explained in Section 3.5, all events are represented as an event node in an event graph. On the other hand, most of the instance level events are not used in composite event expressions. Thus, it is not necessary to create separate primitive event nodes for these instance level events, if they are not used in composition. Thus, we introduce another type of instance level rule which does not require instance level event. In rule definition **r3**, instance IBM is specified along with the class level event ‘setPriceEnd’. In order to accommodate the above instance level rules, an event node contains a special data structure which will be explained in Section 3.5.

### 3.4. Dynamic Creation of Events and Rules

The next issue in the design is to interactively add events and rules, as well as to modify rules. To add events or rules, objects are needed. The objects provided by the application are just the references to the original objects and no duplicates are maintained. Changing the rules at run time would involve the classes containing the rule definition (i.e., dynamically updating the condition part of the ECA Rule). As mentioned earlier, the condition is represented as a method, and another representation is required since any update to the method is reflected only after recompiling the application. Conditions that have attribute comparisons can also be represented as a string instead of a method, so that when the string is evaluated (actually interpreted at runtime) the condition is checked and any update to the string is effective immediately. This representation is flexible and it can also be combined with existing condition methods. The different types of conditions are: attribute-based condition string (simple condition), execution of an existing condition (simple condition) and finally a combination of the above two types (complex condition).

Consider an example class *Track* with two attributes: *Speed* and *Altitude* and a condition based on these attributes represented as “Speed > 700 && Altitude < 20000”. This condition needs to be evaluated at runtime where the values of the attributes “Speed” and “Altitude” have to be substituted following which the resulting string is evaluated. “FESI (Free EcmaScript Interpreter)” [14], a Java-based interpreter is used for the above evaluation. Also, using this approach a complex condition can be created which could contain many simple condition strings connected by Boolean operators. All the objects corresponding to the condition are contained in a *generic* class. Each rule created will contain an instance of this generic class as the condition instance. The usage of the generic class provides the flexibility to change the condition of rule later without having to register the updated condition with the event detector since only the data members are changed and no new instance is created.

### 3.5. Event Graph

The relationship between events and rules is established by means of a subscription and notification mechanism. It is represented by an event graph whose leaf nodes are primitive events and internal nodes are composite events. Every node in the event graph has a list of event subscribers and a list of rule subscribers, where the latter contains references to the rule objects that denote the rules defined for that event and the former contains references to the event nodes of those composite events that have

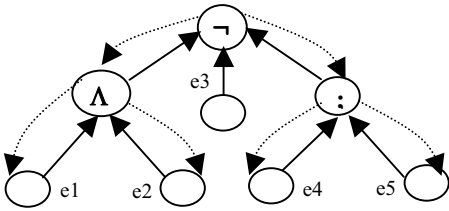


Figure 1. An example event graph

subscribed to that event. A composite event subscribes to the event nodes of all its constituent events. Figure 1 shows the event graph corresponding to the event:

**EVENT notEvent = NOT ((e1 Λ e2), e3, (e4 ; e5))**

A primitive event node is created for every *named* primitive event (class or instance level). It contains the primitive event name, method signature and the event modifier. In order to accommodate the unnamed instance level rules described earlier, a primitive event node contains an instance-rule list that contains a list of instances and a list of rules associated with each instance. As shown in Figure 2, class level rules are associated with a NULL instance and instance level rules are associated with the instance. Storing all the instance level rules in the same primitive event node reduces the overhead of creating multiple event nodes. The event nodes are not necessary when events are used only for defining rules but not in any composite event expression. In the nodes for named events, the instance-rule list contains only a single instance and the set of rules defined on this instance level event. The rule list stores the name of the rule, the references to the condition and action methods, the context, coupling mode, and the priority of the rule. It also stores a rule enable/disable flag that is used to indicate whether the rule is enabled or not.

3.6. Comparing C++ versus Java Design

In the previous sections, we have pointed out several limitations of C++ and how we have overcome those. Below we discuss additional issues.

In C++, condition and action parts of the rule were modeled as functions since C functions can be referenced via function pointers, whereas member functions cannot be referenced. As Java supports references to class methods, conditions and actions can be implemented as methods in Java. In an object-oriented environment, it would be more useful to execute conditions and actions on a class or an object rather than as stand-alone functions. That way, conditions and actions can access the attributes defined in the class.

Java provides the Vector and Hash table data types for storing a collection of

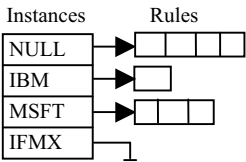


Figure 2. A rule instance list

elements. A Vector is a linear list that can store heterogeneous types of elements. It is more efficient than a linked list used in C++ since the elements can be accessed through an index whereas the elements in a linked list can only be accessed serially. It is particularly more efficient if there are more additions to the Vector than deletions. They are used for storing the list of event subscribers and the list of rules at every event node, and the list of parameter lists at every composite event node. On the other hand, the Hash table gives a faster look-up speed when compared to searching for a particular value in a linked list. It is used to store the mapping of event names and event nodes, event signatures and event nodes and rule names and rule nodes. This is because the references to the event nodes and rule nodes do not change and the references can be uniquely associated with a key such as the event name, event signature or the name of a rule.

#### 4. Implementation of Event Detector

This section describes the implementation details of the event detector.

##### 4.1. Primitive Event Detection

Primitive events are defined using the ‘createPrimitiveEvent’ API which creates a node in the event graph. When a primitive event is defined using the above API, an **event handle** corresponding to that event is returned. The event handle is used to store the parameters of the event as well as to signal the method invocation to the event detector. For example, the below method creates the event ‘setPriceBegin’ with a ‘begin’ event modifier for the method ‘setPrice’ defined in the class ‘Stock’.

```
createPrimitiveEvent (“setPriceBegin”, “Stock”, EventModifier.BEGIN, “void  
setPrice(float)”)
```

In Java, the statements enclosed within the *static* block are executed at the class load time. Rules on the primitive events that belong to a class can be either defined in a *static* block within the class or anywhere in the application. Initially the application invokes the ‘initializeAgent’ method that returns an instance of the ‘ECAAgent’ class. This instance stores the names of all events and their associated handles in a Hash table. The event detector maintains two more hash tables – one stores the mapping between event names and event nodes and the other stores the mapping between method signatures (only for primitive events) and the event nodes. Inside the method that is defined as a primitive event, the user adds calls to the event detector API in order to signal the invocation of a method to the event detector. First, event handles corresponding to the primitive event are obtained using the event name, and the arguments of the method are inserted into all the event handles. The parameters are inserted by specifying the event handle, a symbolic name of the parameter and the parameter itself. Finally, event handles and the instance which invoked the method (*this*) are passed through the ‘raiseBeginEvent’ or ‘raiseEndEvent’ API.

Event detector (Figure 3) uses the event signature to get the corresponding event node from the hash table. Then, the node is notified about the occurrence of the primitive event and the parameter list stored in the event handle is passed to the node as an argument to the ‘notifyEvent’ call. Thus, the invocation of an application method (begin or end) is detected as a primitive event by the event detector. After the primitive event is detected, its parameters are propagated to all the composite events that have

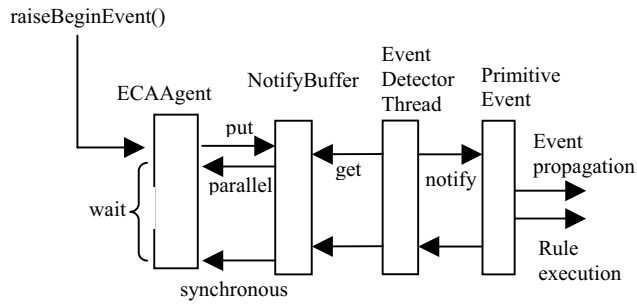


Figure 3. The event detector thread

subscribed to it. When a primitive event occurs the corresponding class level node is notified, the instance rule list is traversed and checked to see if the event instance (the instance which invoked the event method) is present. If the event instance is present, the list of rules associated with the instance is traversed, and the rules are collected for execution.

4.2. Composite Event Detection

Every composite event has an initiator event that initiates the detection, and a terminator event that completes the detection of the event. The composite event is detected when the terminator event is detected. A composite event is detected only when there are rules defined on that event or when rules are defined on another composite event for which this event is a constituent event. For this purpose, there are four numbers stored at each node. Each number denotes the sum of the rules defined on that event including all the dependent events in a particular context. A composite event is detected and propagated in a context only when the corresponding number is non-zero. Whenever a rule is defined or enabled on an event in a particular context, the corresponding integer in that node is incremented. Also, the event detector graph (Figure 4) is recursively traversed from that event node until the leaf nodes are reached

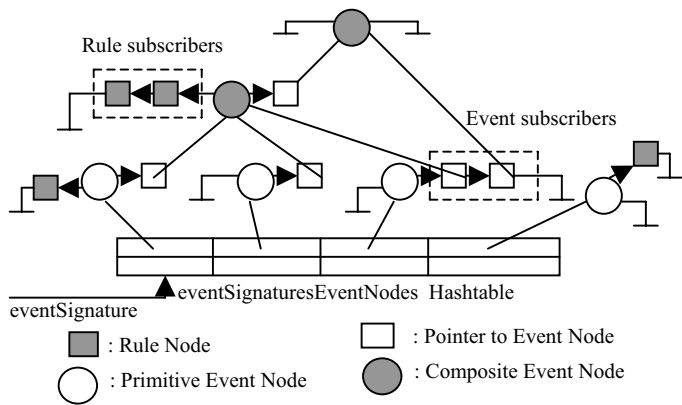


Figure 4. An event detector graph



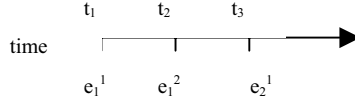


Figure 5. Time line

and the corresponding number is incremented in all the nodes encountered. When a rule is deleted or disabled, the same procedure is followed and the corresponding number is decremented.

A composite event can be detected in four different contexts. To briefly illustrate this, consider a composite event **EVENT andEvent = AND (e1, e2)** and consider the event occurrences shown in the timeline (Figure 5). This event is detected when  $e_2^1$  occurs. Based on the parameter context, event  $e_2^1$  is paired with either  $e_1^1$  or  $e_1^2$  or both. For this example,  $e_1^2$  and  $e_2^1$  are paired in the *recent* context. Events  $e_1^1$  and  $e_2^1$  are paired in the *chronicle* context. In the *continuous* context, two events are detected at the same time –  $e_1^1 e_2^1$  and  $e_1^2 e_2^1$ . In the *cumulative* context, a single event is detected with constituent events  $e_1^1 e_1^2 e_2^1$ .

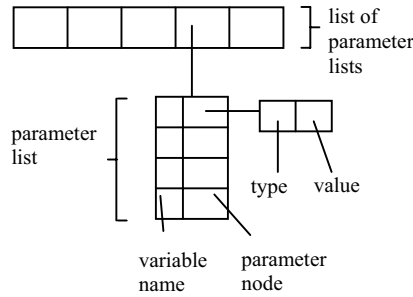
To accomplish the above, the following procedure is used to detect composite events. In every composite event node, an **event table** is stored for each constituent event. Table 1 shows an event table consisting of a set of event entries. Each entry stores an event occurrence and a set of four bits. The event occurrence could be either a single event (a single parameter list) or a set of events (a list of parameter lists). At the first level of composite event nodes in the event graph, the event occurrence is a single event. At composite event nodes higher up in the graph, the event occurrence is a set of events and the cardinality of the set increases as we move higher in the graph. The four bits associated with an event occurrence denote the four contexts – *Recent*, *Chronicle*, *Continuous*, and *Cumulative* in that order. If a particular bit is set, it means that this occurrence of the event is yet to participate in the detection of the composite event in the corresponding context. When an event occurrence is used to detect the composite event in a particular context, the corresponding context bit is reset according to the semantics of the operator.

#### 4.3. Parameter Lists

If the parameters (formal arguments) are inserted into the parameter list in the same order as they are passed to the method, the position of the parameter would be its position in the method definition. When parameters are inserted into a parameter list, they can be retrieved using either the name of the parameter or the position of the parameter in the parameter list. If the parameters are to be retrieved using their name,

Table 1. An event table

$e_1^1 e_2^1 e_3^1$	0010
$e_1^1 e_2^2 e_3^1$	0110
$e_1^1 e_2^1 e_3^2$	1110
$e_1^1 e_2^2 e_3^2$	0001



**Figure 6.** Parameter lists - data structure

both the parameter and its name are to be stored whereas if they are retrieved by their position only, only the parameter needs to be stored. As shown in the Figure 6, a parameter node stores the type and value of the parameter, where the latter is stored as an Object type and the former is needed in order to cast the Object to the appropriate type at run time. A vector, the individual parameter lists, can be accessed both by position as well as sequentially.

#### 4.4. Detecting Temporal Events

Temporal event detection requires a mechanism to keep track of the clock time from within the program. The timer maintained by the operating system could be used for this purpose. An important characteristic of temporal event detection is that the time notifications from the operating system should be asynchronous to the execution of the program. The program should be notified at the moment the designated time expires, irrespective of the execution point in the program. As part of the OS independence, the Java programming language does not provide a mechanism to access the system timer and catch the signals generated by the operating system. Therefore it is not possible to set the system timer and receive notifications from it. But, Java provides a ‘sleep’ method call that causes the calling thread to sleep for a specified amount of time. The sleep method call uses the timer of the underlying operating system to count the time. In Java (unlike in C), the termination of a sleep call from one thread does not cause another sleeping thread to terminate its sleep. However, it is possible to interrupt a sleeping thread from another thread, by calling the interrupt method on the sleeping thread. Thus, when a Java thread calls sleep, it is either terminated at the end of the sleep period or when another thread interrupts it. For the above reasons, the sleep method call is used to implement timer notifications in our system.

A Timer thread is implemented that simulates a timer to detect both absolute and relative temporal events. It runs in an infinite loop sleeping for a certain period of time and then sending a notification to the corresponding temporal event node at the end of the sleep time. When the application defines a temporal event (absolute or relative), a temporal event node is created containing the time expression specified in the temporal event. For absolute events, the timer thread sleeps for the difference amount of the time specified in the absolute temporal event and the current system time. For relative events, the relative time expression is first converted to absolute time and the timer thread is put to sleep for a period that corresponds to the difference between this

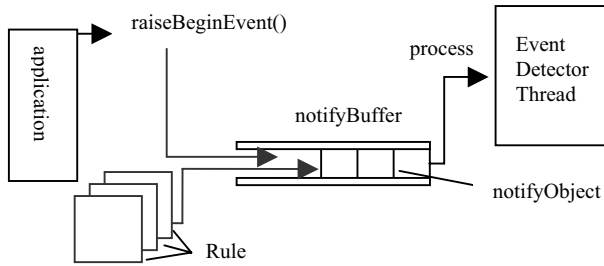
absolute time and the current system time. As there can be more than one temporal event specified and the same clock time may raise multiple events, there should be a way to manage all the temporal events appropriately. While the timer thread is sleeping for a certain period of time, there could be another definition of a temporal event that occurs before the current sleep time. For this purpose, the timer thread should be able to be interrupted while it is sleeping, and then put to sleep again for a different amount of time. The Temporal Event Handler component of the event detector takes care of all these by managing the temporal events appropriately.

#### 4.4.1. Temporal Event Handler

The temporal event handler has three classes – *TimeItem*, *TimeQueue* and *Timer*. The *TimeItem* class stores a time expression and an *event\_id* that refers to the precedent event for relative time events. An absolute event has the format – hh:mm:ss/MM/dd/yyyy and supports the specification of wild cards '?' and '\*' in one or more positions of the absolute time expression. The implementation of repetitive temporal events follows the algorithm specified in [15]. A relative time expression appears in the definition of one of the temporal events – PLUS, P or P\*. It has the format – hh 'hrs' mm 'min' ss 'sec'. The relative time is converted to an absolute time by adding the relative time to the system time when the *TimeItem* is instantiated. A *TimeQueue* is a linked list of *TimeItems* maintained in the ascending order of their time values and there are two *TimeQueues* namely, *allItems* and *presentItems*. The list of *presentItems* consists of all the time items that are to be notified at the same time. Initially, the incoming time item is put in the *presentItems* list and the timer thread is put to sleep accordingly. When a new time item comes while the timer thread is sleeping, the time value in the new time item may be lesser, greater or equal to the time value for which the timer thread is currently sleeping. In the first case, the new time item is added to the list of *presentItems*. In the second case, all the time items in the *presentItems* list are moved to the list of *allItems* and the new time value is placed in the *presentItems* list. The timer thread is interrupted and put to sleep for the new time value. In the third case the new time value is placed in the *allItems* list in the appropriate position. It is to be remembered that the *allItems* list is maintained in the ascending order of time values. Whenever the time items in the *presentItems* list are processed, all the time items in the *allItems* list that have the same time value are moved to the list of *presentItems* and the timer thread is put to sleep according to the new time. The timer thread runs in an infinite loop, checking for items present in the *presentItems* list that are to be notified.

#### 4.5. Event Detector as a Thread

Event detector as a thread is shown in Figure 7. Java implementation, unlike C++, separates the detector and the application into two different threads. This allows the detection of events to be processed either synchronously or in parallel with the application. Whenever the application makes a call to the *raiseBeginEvent* API, a reference to that method is constructed and placed in an object. The event detector runs as a thread in an infinite loop that continuously keeps getting objects from the *notifyBuffer* and executing the method calls stored in those objects with the parameters stored in that object. Only the *raiseBegin/EndEvent* calls are put into the *notifyBuffer*



**Figure 7.** Processing event notifications from event detector thread

and processed by the event detector thread. The event detector thread does not process the calls to the API methods for creating events and rules. This is because the API for creating events returns an event handle, which is not possible if the event detector thread processes these calls, since threads cannot return a value.

## 5. Conclusions

This paper presents the design, architecture, and implementation aspects of incorporating active capability into a Java application environment. An event detector has been implemented that detects events in Java applications and executes rules defined on them. It also allows for creating and modifying composite and temporal events and rules dynamically. The system runs on NT, Solaris, and Linux Operating systems providing true portability. This paper contrasts the limitations of the C++ environment with that of Java environment and describes how we have overcome the limitations. A number of applications have been implemented using this prototype. Details of algorithms for event detection and application development can be found in [16]. In the C++ version, there was only a single event graph for all the events and rules in an application whereas the current implementation allows multiple event graphs, each associated with a set of events and rules. This allows grouping of events and rules within the same application. By grouping rules, the application can disable or enable a group of rules and also use different groups of rules on the same events in different parts of the application.

## References

- [1] Anwar, E., L. Maugis, and S. Chakravarthy, A New Perspective on Rule Support for Object-Oriented Databases. In: Proc. of the 1993 ACM SIGMOD Int'l Conference on Management of Data. 1993, Washington D.C., 99-108.
- [2] Chakravarthy, S. and D. Mishra, Snoop: an Expressive Event Specification Language for Active Databases. *Data and Knowledge Engineering*, 14(10), 1994, 1-26.
- [3] Adaikkalavan, R. and S. Chakravarthy, SnoopIB: Interval-Based Event Specification and Detection for Active Databases. *Data and Knowledge Engineering (DKE)*, 59(1), 2006, Elsevier, 139-165.
- [4] Krishnaprasad, V., Event Detection for Supporting Active Capability in an OODBMS: Semantics, Architecture, and Implementation. MS Thesis, 1994, CIS Department, University of Florida, Gainesville.
- [5] Gehani, N.H., H.V. Jagadish, and O. Shmueli, COMPOSE: a System for Composite Event Specification and Detection. 1992, AT&T Bell Laboratories.

- [6] Gehani, N. and H.V. Jagadish, Ode as an Active Database: Constraints and Triggers. In: Proc. 17th Int'l Conf. on Very Large Data Bases VLDB, 1991, Barcelona, Spain, 327-336.
- [7] Diaz, O., N. Paton, and P. Gray, Rule Management in Object-Oriented Databases: a Unified Approach. In: Proc. 17th Int'l Conf. on Very Large Data Bases VLDB, 1991, Barcelona, Spain, 317-326.
- [8] Gatzia, S. and K.R. Dittrich, Events in an Active Object-Oriented System. In: Rules in Database Systems, N. Paton and M. Williams, Eds, 1993, Springer, 127-142.
- [9] Gatzia, S. and K.R. Dittrich, Detecting Composite Events in Active Databases Using Petri Nets. In: Proc. of the 4th Int'l Workshop on Research Issues in Data Engineering: Active Database Systems, 1994, 2-9.
- [10] Chakravarthy, S., et al., Composite Events for Active Databases: Semantics, Contexts and Detection. In: Proc. Int'l. Conf. on Very Large Data Bases VLDB, 1994, Santiago, Chile, 606-617.
- [11] Chakravarthy, S., et al., Design of Sentinel: an Object-Oriented DBMS with Event-Based Rules. Information and Software Technology, 36(9), 1994, 559-568.
- [12] Vitria BusinessWare. Web site [cited Oct 2006]. Available from: <http://www.vitria.com>.
- [13] WebLogic Events Architecture. Web site [cited Oct 2006]. Available from: <http://www.weblogic.com/docs/techoverview/em.html>.
- [14] Lugin, J.-M., Free EcmaScript Interpreter: a JavaScript Interpreter Written in Java. 2003. [cited Oct 2006]. Available from: <http://www.lugin.ch/fesi/interp.html>.
- [15] A.Ziemann, G., Detailed Design Description: Time Queue Handler. 1995, University of Florida, Gainesville.
- [16] Dasari, R., Events and Rules for JAVA: Design and Implementation of a Seamless Approach. MS Thesis, 1999, Database Systems R&D Center, CIS Department, University of Florida, Gainesville.

# On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models

Giancarlo GUIZZARDI

*Federal University of Espírito Santo (UFES), Vitória, Brazil*

*Laboratory for Applied Ontology (ISTC-CNR), Trento, Italy*

*guizzardi@loa-cnr.it*

**Abstract.** In philosophy, the term ontology has been used since the 17th century to refer both to a philosophical discipline (Ontology with a capital “O”), and as a domain-independent system of categories that can be used in the conceptualization of domain-specific scientific theories. In the past decades there has been a growing interest in the subject of ontology in computer and information sciences. In the last few years, this interest has expanded considerably in the context of the Semantic Web and MDA (Model-Driven Architecture) research efforts, and due to the role ontologies are perceived to play in these initiatives. In this paper, we explore the relations between Ontology and ontologies in the philosophical sense with domain ontologies in computer science. Moreover, we elaborate on formal characterizations for the notions of ontology, conceptualization and metamodel, as well as on the relations between these notions. Additionally, we discuss a set of criteria that a modeling language should meet in order to be considered a suitable language to model phenomena in a given domain, and present a systematic framework for language evaluation and design. Furthermore, we argue for the importance of ontology in both philosophical senses aforementioned for designing and evaluating a suitable general ontology representation language, and we address the question whether the so-called Ontology Web languages can be considered as suitable general ontology representation languages. Finally, we motivate the need for two complementary classes of modeling languages in Ontology Engineering addressing two separate sets of concerns.

**Keywords.** Formal ontology, conceptual modeling, metamodeling, language adequacy

## Introduction

The Webster dictionary [1] defines the term ontology as:

- (D1). a branch of metaphysics concerned with the nature and relations of being;
- (D2). a particular theory about the nature of being or the kinds of existents;
- (D3). a theory concerning the kinds of entities and specifically the kinds of

abstract entities that are to be admitted to a language system.

Etymologically, *ont-* comes from the present participle of the Greek verb *einai* (to be) and, thus, the latin word *Ontologia* (*ont-* + *logia*) can be translated as *the study of existence*. The term was coined in the 17<sup>th</sup> century in parallel by the philosophers [Rudolf Göckel](#) in his *Lexicon philosophicum* and by Jacob Lorhard in his *Ogdoas Scholastica*, but popularized in philosophical circles only in 18<sup>th</sup> century with the

publication in 1730 of the *Philosophia prima sive Ontologia* by the German philosopher Christian Wolff.

In the sense (D1) above, ontology is the most fundamental branch of metaphysics. Aristotle was the first western philosopher to study metaphysics systematically and to lay out a rigorous account of ontology. He described (in his *Metaphysics* and *Categories*) ontology as *the science of being qua being*. According to this view, the business of ontology is to study the most general features of reality and real objects, i.e., the study of the generic traits of every mode of being. As opposed to the several specific scientific disciplines (e.g., physics, chemistry, biology), which deal only with entities that fall within their respective domain, ontology deals with transcategorical relations, including those relations holding between entities belonging to distinct domains of science, and also by entities recognized by *common sense*.

In the beginning of the 20<sup>th</sup> century the German philosopher Edmund Husserl coined the term *Formal Ontology* as an analogy to Formal Logic. Whilst Formal Logic deals with formal logical structures (e.g., truth, validity, consistency) independently of their veracity, Formal Ontology deals with formal ontological structures (e.g., theory of parts, theory of wholes, types and instantiation, identity, dependence, unity), i.e., with formal aspects of objects irrespective of their particular nature. The unfolding of Formal Ontology as a philosophical discipline aims at developing a system of general categories and their ties, which can be used in the development of scientific theories and domain-specific common sense theories of reality. In other words, Ontology (as a discipline, with capital O) in the first sense of Webster's definition aforementioned contributes to the development of ontologies in the second sense. The first ontology developed in sense (D2) is the set of theories of Substance and Accidents developed by Aristotle in his *Metaphysics* and *Categories*. Since then, ontological theories have been proposed by innumerable authors in philosophy, and more recently also in the area of *Applied Ontology* in computer science (e.g., DOLCE, GFO, OCCHRE, UFO).

The term "*ontology*" in the computer and information science literature appeared for the first time in 1967, in a work on the foundations of data modeling by S. H. Mealy, in a passage where he distinguishes three distinct realms in the field of data processing, namely: (i) "*the real world itself*"; (ii) "*ideas about it existing in the minds of men*"; (iii) "*symbols on paper or some other storage medium*". Mealy concludes the passage arguing about the existence of things in the world regardless of their (possibly) multiple representations and claiming that "*This is an issue of ontology, or the question of what exists*" [2,p.525]. In the end of this passage, Mealy includes a reference to Quine's essay "*On What There Is*" [3]. In an independent manner, yet another sub-field of computer science, namely Artificial Intelligence (AI) began to make use of what came to be known as *domain ontologies*. Since the first time the term was used in AI by Hayes [4] and since the development of his naïve physics ontology of liquids [5], a large amount of domain ontologies have been developed in a multitude of subject areas. In the past five years, an explosion of works related to ontology has happened in computer science, chiefly motivated by the growing interest on the Semantic Web, and by the key role played by them in that initiative.

An important point that should be emphasized is the difference in the senses of the term used by the information systems, on one side, and artificial intelligence and semantic web communities on the other. In information systems, the term ontology has been used in ways that conform to its definitions in philosophy (in both senses D1 and D2). As a system of categories, an ontology is independent of language: Aristotle's ontology is the same whether it is represented in English, Greek or First-Order Logic.

In contrast, in most of other areas of computer science (the two latter areas included), the term ontology is, in general, used as a concrete engineering artifact designed for a specific purpose, and represented in a specific language.

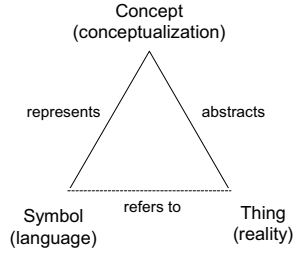
In the light of these contrasting notions of ontologies, a number of question begging issues become manifest: What exactly is a domain ontology? How does it relate to other concrete representations such as conceptual models and metamodels? How does it relate to ontology in the philosophical senses (D1-D3) aforementioned? Additionally, during the years many languages have been used to represent domain ontologies. Examples include Predicate Calculus, KIF, Ontolingua, UML, EER, LINGO, ORM, CML, DAML+OIL, F-Logic, OWL. What are the characteristics that a suitable language to represent conceptual models, in general, and domain ontologies, in particular should have? In particular, are the semantic web languages suitable ontology representation languages?

The objective of this article is to offer answers to these questions. In the next section, we start by discussing the relation between reality, conceptualization and language, and by briefly introducing a framework that can be used to systematically evaluate and re-design a modeling language w.r.t. its suitability to model phenomena in a given domain. In Section 2, we elaborate on the notion of a language metamodel. In Section 3, we provide a formal account for the notion of domain ontology as well as for its relation to conceptualization and language metamodel as discussed in Section 1. In Section 4, we advocate the need for an ontologically well-founded system of categories that should underlie a suitable ontology representation language (i.e., an ontology in the sense D2 and D3), and discuss the role played by Formal Ontology in Philosophy (Ontology in the sense D1) in the development of such a system. In Section 5, we make use of the framework of Section 1 and provide a concrete example to illustrate many of the notions discussed in the article, namely, those of foundational and domain ontology, ontology representation language, domain-specific language, and (meta)model. In Section 6, we motivate the need for two complementary classes of representation languages in Ontology Engineering: one class populated by philosophically well-founded languages, focused on expressivity and conceptual clarity, and another one populated by languages focused on computation-oriented concerns (e.g., decidability, efficient automated reasoning, etc.). In Section 7, we conclude this article with a summary of the most important points discussed herein.

## 1. Conceptualization and Language

One of the main success factors behind the use of a modeling language lies in the language's ability to provide to its target users a set of modeling primitives that can directly express relevant domain concepts, comprising what we name here a *domain conceptualization*. The elements constituting a *conceptualization* of a given domain are used to articulate abstractions of certain state of affairs in reality. We name the latter *domain abstractions*. Take as an example the domain of genealogical relations in reality. A certain conceptualization of this domain can be constructed by considering concepts such as *Person*, *Man*, *Woman*, *Father*, *Mother*, *Offspring*, *being the father of*, *being the mother of*, among others. By using these concepts, we can articulate a domain abstraction (i.e., a mental model) of certain facts in reality such as, for instance, that *a man named John is the father of another man named Paul*.



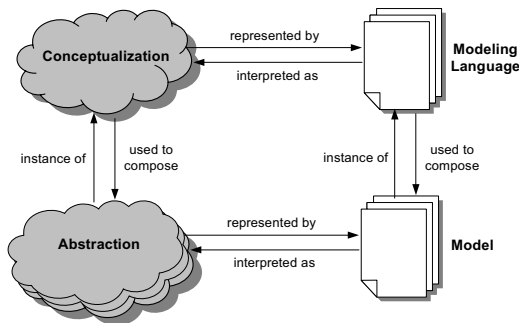


**Figure 1.** Ullmann's Triangle: the relations between a thing in reality, its conceptualization and a symbolic representation of this conceptualization.

Conceptualizations and Abstractions are immaterial entities that only exist in the mind of the user or a community of users of a language. In order to be documented, communicated and analyzed they must be captured, i.e. represented in terms of some concrete artifact. This implies that a language is necessary for representing them in a concise, complete and unambiguous way. Figure 1 represents the relation between a language, a conceptualization and the portion of reality that this conceptualization abstracts. This picture depicts the well-known *Ullmann's triangle* [6]. This triangle derives from that of Ogden and Richards [7] and from Ferdinand de Saussure [8], on whose theories practically the whole modern science of language is based.

The *represents* relation concerns the definition of language  $\mathcal{L}$ 's *real-world semantics*. The dotted line between language and reality in this figure highlights the fact that the relation between language and reality is always intermediated by a certain conceptualization [9]. This relation is elaborated in Figure 2 that depicts the distinction between an abstraction and its representation, and their relationship with conceptualization and representation language. In the scope of this work the representation of a domain abstraction in terms of a representation language  $\mathcal{L}$  is called a *model specification* (or simply *model*, *specification*, or *representation*) and the language  $\mathcal{L}$  used for its creation is called a *modeling* (or *specification*) *language*.

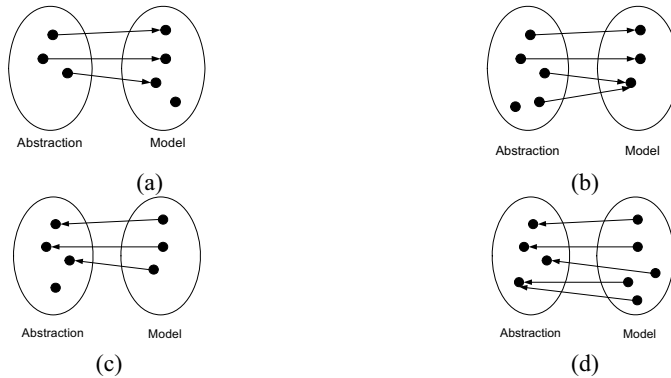
In order for a model  $\mathcal{M}$  to faithfully represent an abstraction  $\mathcal{A}$ , the modeling primitives of the language  $\mathcal{L}$  used to produce  $\mathcal{M}$  should faithfully represent the domain conceptualization  $\mathcal{C}$  used to articulate the represented abstraction  $\mathcal{A}$ . The *Domain Appropriateness* of a language is a measure of the suitability of a language to model phenomena in a given domain, or in other words, of its truthfulness of a language to a



**Figure 2.** Relations between Conceptualization, Abstraction, Modeling Language and Model

given domain in reality. On a different aspect, different languages and specifications have different measures of pragmatic adequacy [10]. *Comprehensibility appropriateness* refers to how easy is for a user a given language to recognize what that language's constructs mean in terms of domain concepts and, how easy is to understand, communicate and reason with the specifications produced in that language. The measures of these two quality criteria for a given language and domain are aspects of the *represents* relation depicted in Figure 1, and they can be systematically evaluated by comparing, on one hand, a concrete representation of the worldview underlying that language (captured by that language's *metamodel*) to, on the other hand, a concrete representation of a domain conceptualization, or a *domain ontology*. The truthfulness to reality (*domain appropriateness*) and conceptual clarity (*comprehensibility appropriateness*) of a modeling language depend on the level of homomorphism between these two entities. The stronger the match between an abstraction in reality and its representing model, the easier is to communicate and reason with that model.

In [10], we discuss a number of properties that should be reinforced for an isomorphic mapping to take place between an ontology  $O$  representing a domain  $\mathcal{D}$  and a domain language's metamodel. If isomorphism can be guaranteed, the implication for the human agent who interprets a diagram (model) is that his interpretation correlates precisely and uniquely with an abstraction being represented. By contrast, where the correlation is not an isomorphism then there may potentially be a number of unintended abstractions which would match the interpretation. These properties are briefly discussed in the sequel and are illustrated in Figure 3: (a) *Soundness*: A language  $\mathcal{L}$  is *sound* w.r.t. to a domain  $\mathcal{D}$  iff every modeling primitive in the language has an interpretation in terms of a domain concept in the ontology  $O$ ; (b) *Completeness*: A language  $\mathcal{L}$  is *complete* w.r.t. to a domain  $\mathcal{D}$  iff every concept in the ontology  $O$  of that domain is represented in a modeling primitive of that language; (c) *Lucidity*: A language  $\mathcal{L}$  is *lucid* w.r.t. to a domain  $\mathcal{D}$  iff every modeling primitive in the language represents at most one domain concept in  $O$ . (d) *Laconicity*: A language  $\mathcal{L}$  is *laconic* w.r.t. to a domain  $\mathcal{D}$  iff every concept in the ontology  $O$  of that domain is represented at most once in the metamodel of that language. In the same article, we also provide a methodological framework for assessing these properties given a language and a domain. Such framework has been applied in a number of case studies. The most



**Figure 3.** Examples of Lucid (a) and Sound (b) representational mappings from Abstraction to Model; Examples of Laconic (c) and Complete (d) interpretation mappings from Model to Abstraction.

comprehensive example being [10] with the evaluation and re-design of UML for the purpose of conceptual modeling, but also [11], in which this framework is employed to design an agent-oriented engineering methodology for the domain of Knowledge Management.

*Unsoundness, Non-Lucidity, Non-Laconicity and Incompleteness* violate what the philosopher of language H.P. Grice [12] names *conversational maxims* that states that a speaker is assumed to make contributions in a dialogue which are *relevant, clear, unambiguous*, and *brief, not overly informative and true according to the speaker's knowledge*. Whenever models do not adhere to these conversational maxims, *they* can communicate incorrect information and induce the user to make incorrect inferences about the semantics of the domain.

## 2. Language and Metamodel

The set of symbols that compose a language as well as the rules for forming valid combinations of these symbols constitute the language's syntax. In sentential languages, the syntax is first defined in terms of an alphabet (set of characters) that can be grouped into valid sequences forming words. This is called a lexical layer and it is typically defined using regular expressions. Words can be grouped into sentences according to precisely defined rules defined in a context-free grammar, resulting in an abstract syntax tree. Finally, these sentences are constrained by given context conditions.

In diagrammatic (graphical) languages, conversely, the syntax is not defined in terms of linear sequence of characters but in terms of pictorial signs. The set of available graphic modeling primitives forms the lexical layer and the language abstract syntax is typically defined in terms of a *metamodel*. Finally, the language metamodel is enriched by context-conditions given in some constraint description language, such as, OCL or first-order logic (FOL). In either case, context conditions are intended to constrain the language syntax by defining the set of correct (well-formed) sentences of the language. Some of these constraints are motivated by semantic considerations (laws of the domain being modeled as we shall see) while others are motivated by pragmatic issues [10]. Nevertheless, a metamodel is a description of the language's abstract syntax since it defines: (i) a set of constructs selected for the purpose of performing a specific (set of) task(s) and, (ii) a set of well-formedness rules for combining these constructs in order to create grammatically valid models in the language.

In the previous section, we advocate that the suitability of a language to create models in a given domain depends on how "*close*" the structure of the models constructed using that language resemble the structure of the domain abstractions they are supposed to represent. To put it more technically, a model  $\mathcal{M}$  produced in a language  $\mathcal{L}$  should be, at least, a homomorphism of the abstraction  $\mathcal{A}$  that  $\mathcal{M}$  represents. This evaluation can be systematically performed ultimately based on the analysis of the relation between the structure of a modeling language and the structure of a domain conceptualization.

What is referred by *structure of a language* can be accessed via the description of the specification of *conceptual model underlying the language*, i.e., a description of the worldview embedded in the language's modeling primitives. In [13], this is called the *ontological metamodel of the language*, or simply, the *ontology of the language*. From a philosophical standpoint, this view is strongly associated with Quine [14], who proposes that an ontology can be found in the *ontological commitment* of a given

language, that is, the entities the primitives of a language commit to the existence of. For example, Peter Chen's Entity Relationship model [15] commits to a worldview that accounts for the existence of three types of things: *entity*, *relationship* and *attribute*.

This distinction of a metamodel as a pure description of a language's abstract syntax and as a description of the worldview underlying the language can be understood in analogy to the distinction between a design model and a conceptual model in information systems and software engineering. Whilst the latter is only concerned with modeling a view of the domain for a given application (or class of applications), the former is committed to translating the model of this view on the most suitable implementation according to the underlying implementation environment and also considering a number of non-functional requirements (e.g., security, fault-tolerance, adaptability, reusability, etc.). Likewise, the specification of the conceptual model underlying a language is the description of what the *primitives of a language* are able to represent in terms of real-world phenomena. In some sense (formally characterized in the next section), it is the representation of a conceptualization of the domain in terms of the language's vocabulary. In the *design* of a language, these conceptual primitives can be translated into a different set of primitives. For example, it can be the case that a conceptual primitive is not directly represented in the actual abstract syntax of a language, but its modeling capabilities (the real world concept underlying it) can be translated to several different elements in the language's abstract syntax due to non-functional requirements (e.g., pragmatics, efficiency). Nonetheless, the design of a language is responsible for guaranteeing that the language's syntax, formal semantics and pragmatics are conformant with this conceptual model. From now on, the *Modeling Language* icon depicted in Figure 2 represents the specification of the conceptual model underlying the language, or what we shall name the *language metamodel specification*, or simply the *language metamodel*.

The *structure of domain conceptualization* must also be made accessible through an explicit and formal description of the corresponding portion of reality in terms of a concrete artifact, which is termed here a *domain reference ontology*, or simply, a *domain ontology*. The idea is that a reference ontology should be constructed with the sole objective of making the best possible description of the domain in reality w.r.t. to a certain level of granularity and viewpoint. The notion of ontology as well as its role in the explicit representation of conceptualizations is discussed in depth and given a formal characterization in the next section.

### 3. Ontology, Metamodel and Conceptualization

Let us now return our attention to Figure 2. A Modeling language can be seen as delimiting all possible specifications<sup>1</sup> which can be constructed using that language, i.e., all grammatically valid specifications of that language. Likewise, a conceptualization can be seen as delimiting all possible domain abstractions (representing state of affairs) which are admissible in that domain [16]. Therefore, for example, in a conceptualization of the domain of genealogy, there cannot be a domain

---

<sup>1</sup> We have so far used the term *model* instead of specification since it is the most common term in conceptual modeling. In this session, exclusively, we adopt the latter in order to avoid confusion with the term (logical) model as used in logics and tarskian semantics. A specification here is a syntactic notion; a logical model is a semantic one.

abstraction in which a person is his own biological parent, because such a state of affairs cannot happen in reality. Accordingly, we can say that a modeling language which is truthful to this domain is one which has as valid (i.e., grammatically correct) specifications only those that represent state of affairs deemed admissible by a conceptualization of that domain. In the sequel, following [16], we present a formalization of this idea. This formalization compares conceptualizations as intentional structures and metamodels as represented by logical theories.

Let us first define a *conceptualization*  $C$  as follows:

**Definition 1 (conceptualization):** A conceptualization  $C$  is an intensional structure  $\langle W, D, \mathfrak{R} \rangle$  such that  $W$  is a (non-empty) set of possible worlds,  $D$  is the domain of individuals and  $\mathfrak{R}$  is the set of  $n$ -ary relations (concepts) that are considered in  $C$ . The elements  $\rho \in \mathfrak{R}$  are intensional (or conceptual) relations with signatures such as  $\rho^n: W \rightarrow \wp(D^n)$ , so that each  $n$ -ary relation is a function from possible worlds to  $n$ -tuples of individuals in the domain. ■

For instance, we can have  $\rho$  accounting for the meaning of the natural kind apple. In this case, the meaning of apple is captured by the intentional function  $\rho$ , which refers to all instances of apples in every possible world.

**Definition 2 (intended world structure):** For every world  $w \in W$ , according to  $C$  we have an *intended world structure*  $S_w C$  as a structure  $\langle D, R_w C \rangle$  such that  $R_w C = \{\rho(w) \mid \rho \in \mathfrak{R}\}$ . ■

More informally, we can say that every intended world structure  $S_w C$  is the characterization of some state of affairs in world  $w$  deemed admissible by conceptualization  $C$ . From a complementary perspective,  $C$  defines all the admissible state of affairs in that domain, which are represented by the set  $S_c = \{S_w C \mid w \in W\}$ .

Let us consider now a language  $\mathcal{L}$  with a vocabulary  $V$  that contains terms to represent every concept in  $C$ .

**Definition 3 (logical model):** A logical model for  $\mathcal{L}$  can be defined as a structure  $\langle S, I \rangle$ :  $S$  is the structure  $\langle D, R \rangle$ , where  $D$  is the domain of individuals and  $R$  is a set of extensional relations;  $I: V \rightarrow D \cup R$  is an interpretation function assigning elements of  $D$  to constant symbols in  $V$ , and elements of  $R$  to predicate symbols of  $V$ . A model, such as this one, fixes a particular extensional interpretation of language  $\mathcal{L}$ . ■

**Definition 4 (intensional interpretation):** Analogously, we can define an intensional interpretation by means of the structure  $\langle C, \mathfrak{I} \rangle$ , where  $C = \langle W, D, \mathfrak{R} \rangle$  is a conceptualization and  $\mathfrak{I}: V \rightarrow D \cup \mathfrak{R}$  is an intensional interpretation function which assigns elements of  $D$  to constant symbols in  $V$ , and elements of  $\mathfrak{R}$  to predicate symbols in  $V$ . ■

In [16], this intentional structure is named the *ontological commitment* of language  $\mathcal{L}$  to a conceptualization  $C$ . We therefore consider this intensional relation as corresponding to the *represents* relation depicted in Ullmann's triangle in Figure 1.

**Definition 5 (ontological commitment):** Given a logical language  $\mathcal{L}$  with vocabulary  $V$ , an *ontological commitment*  $K = \langle C, \mathfrak{I} \rangle$ , a model  $\langle S, I \rangle$  of  $\mathcal{L}$  is said to be compatible with  $K$  if: (i)  $S \in S_c$ ; (ii) for each constant  $c$ ,  $I(c) = \mathfrak{I}(c)$ ; (iii) there exists a world  $w$  such that for every predicate symbol  $p$ ,  $I$  maps such a predicate to an admissible extension of  $\mathfrak{I}(p)$ , i.e. there is a conceptual relation  $\rho$  such that  $\mathfrak{I}(p) = \rho$  and  $\rho(w) =$

I(p). The set  $I_k(\mathcal{L})$  of all models of  $\mathcal{L}$  that are compatible with  $K$  is named the set of *intended models* of  $\mathcal{L}$  according to  $K$ . ■

**Definition 6 (logical rendering):** Given a specification  $X$  in a specification language  $\mathcal{L}$ , we define as the logical rendering of  $X$ , the logical theory  $T$  that is the first-order logic description of that specification [17]. ■

In order to exemplify these ideas let us take the example of a very simple conceptualization  $C$  such that  $W = \{w, w'\}$ ,  $D = \{a, b, c\}$  and  $\mathcal{R} = \{\text{person}, \text{father}\}$ . Moreover, we have that  $\text{person}(w) = \{a, b, c\}$ ,  $\text{father}(w) = \{a\}$ ,  $\text{person}(w') = \{a, b, c\}$  and  $\text{father}(w') = \{a, b\}$ . This conceptualization accepts two possible state of affairs, which are represented by the world structures  $S_w C = \{\{a, b, c\}, \{\{a, b, c\}, \{a\}\}$  and  $S_{w'} C = \{\{a, b, c\}, \{\{a, b, c\}, \{a, b\}\}$ . Now, let us take a language  $\mathcal{L}$  whose vocabulary is comprised of the terms *Person* and *Father* with an underlying metamodel that poses no restrictions on the use of these primitives. In other words, the metamodel of  $\mathcal{L}$  has the following logical rendering ( $T_1$ ):

1.  $\exists x \text{ Person}(x)$
2.  $\exists x \text{ Father}(x)$

In this case, we can clearly produce a logical model of  $\mathcal{L}$  (i.e., an interpretation that validates the logical rendering of  $\mathcal{L}$ ) but that is not an intended world structure of  $C$ . For instance, the model  $D' = \{a, b, c\}$ ,  $\text{person} = \{a, b\}$ ,  $\text{father} = \{c\}$ , and  $I(\text{Person}) = \text{person}$  and  $I(\text{Father}) = \text{father}$ . This means that we can produce a specification using  $\mathcal{L}$  which model is not an *intended model* according to  $C$ .

Now, let us update the metamodel of language  $\mathcal{L}$  by adding one specific axiom and, hence, producing the metamodel ( $T_2$ ):

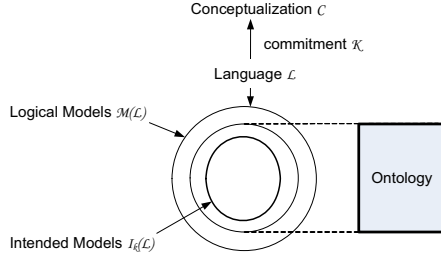
1.  $\exists x \text{ Person}(x)$
2.  $\exists x \text{ Father}(x)$
3.  $\forall x \text{ Father}(x) \rightarrow \text{Person}(x)$

Contrary to  $\mathcal{L}$ , the resulting language  $\mathcal{L}'$  with the amended metamodel  $T_2$  has the desirable property that all its *valid specifications have logical models that are intended world structures of  $C$* .

We can summarize the discussion so far as follows. A domain conceptualization  $C$  can be understood as describing the set of all possible state of affairs, which are considered admissible in a given universe of discourse  $U$ . Let  $V$  be a vocabulary whose terms directly correspond to the intensional relations in  $C$ . Now, let  $X$  be a *conceptual specification* (i.e., a concrete representation) of universe of discourse  $U$  in terms of the vocabulary  $V$  and let  $T_X$  be a logical rendering of  $X$ , such that its axiomatization constrains the possible interpretations of the members of  $V$ . We call  $X$  (and  $T_X$ ) an *ideal ontology* of  $U$  according to  $C$  iff the logical models of  $T_X$  describe all and only state of affairs which are admitted by  $C$ .

The relationships between language vocabulary, conceptualization, ontological commitment and ontology are depicted in Figure 4. This use of the term ontology is strongly related to the third sense (D3) in which the term is used in philosophy, i.e. as “a theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system” (Introduction).

The logical theory ( $T_2$ ) described above is, thus, an example of an ontology for the person/father toy conceptualization. As pointed out in [16], ontologies cannot always be ideal and, hence, a general definition for an (non-ideal) ontology must be given: *An*

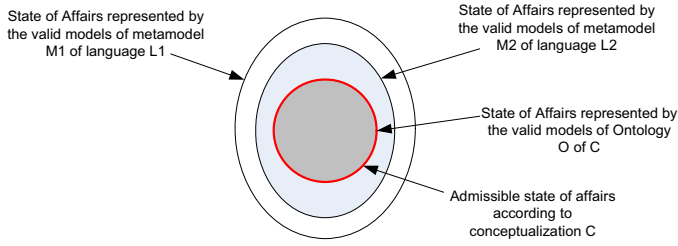


**Figure 4.** Relations between language (vocabulary), conceptualization, ontological commitment and ontology

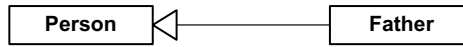
*ontology is a conceptual specification that describes knowledge about a domain in a manner that is independent of epistemic states and state of affairs. Moreover, it intends to constrain the possible interpretations of a language's vocabulary so that its logical models approximate as well as possible the set of intended world structures of a conceptualization  $C$  of that domain.*

According to this criterion of accuracy, we can therefore give a precise account for the quality of a given ontology. Given an ontology  $O_L$  and an ideal ontology  $O_C$ , the quality of  $O_L$  can be measured as the distance between the set of logical models of  $O_L$  and  $O_C$ . In the best case, the two ontologies share the same set of logical models. In particular, if  $O_L$  is the specification of the ontological metamodel of modeling language  $\mathcal{L}$ , we can state that if  $O_L$  and  $O_C$  are *isomorphic* then they also share the same set of possible models. It is important to emphasize the relation between the possible models of  $O_L$  and the completeness of language  $\mathcal{L}$  (in the technical sense briefly discussed in Section 1). There are two ways in which incompleteness can impact the quality of  $O_L$ : firstly, if  $O_L$  (and thus  $\mathcal{L}$ ) does not contain concepts to fully characterize a state of affairs, it is possible that the logical models of  $O_L$  describe situations that are present in several world structures of  $C$ . In this case,  $O_L$  is said to *weakly* characterize  $C$  [16], since it cannot guarantee the reconstruction of the relation between worlds and extensional relations established by  $C$ ; secondly, if the representation of a concept in  $O_L$  is underspecified, it will not contain the axiomatization necessary to exclude unintended logical models. As an example of the latter, we can mention the incompleteness of UML class diagrams w.r.t. classifiers and part-whole relations discussed in [10]. In summary, we can state that an ideal ontology  $O_C$  for a conceptualization  $C$  of universe of discourse  $U$  can be seen as the specification of the ontological metamodel for an ideal language to represent  $U$  according to  $C$ . For this reason, the adequacy of a language  $\mathcal{L}$  to represent phenomena in  $U$  can be systematically evaluated by comparing  $\mathcal{L}$ 's metamodel specification with  $O_C$ . This idea is illustrated in Figure 5.

By including formula (3),  $(T_1)$  is transformed into an ideal ontology  $(T_2)$  of  $C$ . One question that comes to the mind is: *How can one know that?* In other words, how can we systematically design an ontology  $O$  that is a better characterization of  $C$ . There are two important points that should be called to attention. The first point concerns the language that is used in the representation of these specifications, namely, that of standard predicate calculus. The formula added to  $(T_1)$  to create  $(T_2)$  represents a subsumption relation between Person and Father. Subsumption is a basic primitive in the group of the so-called *epistemological languages*, which includes languages such as



**Figure 5.** Measuring the degree of *domain appropriateness* of modeling languages via an ontology of a conceptualization of that domain.



**Figure 6.** Example of a subsumption relation in UML

UML, EER and OWL. It is, in contrast, absent in ontological neutral logical languages such as predicate calculus. By using a language such as OWL to represent a conceptualization of this domain, a specification such as the one in Figure 6 should be produced (which reads “Father *is-a* Person”, or in other words, the Father concept is subsumed by the Person concept). In this model, the third axiom would be automatically included through the semantics of the metamodeling language. Therefore, if a suitable ontology modeling language is chosen, its primitives incorporate an axiomatization, such that the specifications (ontologies) produced using this language will better approximate the intended models of a conceptualization C.

The second point that should be emphasized is related to the question: *how are the world structures that are admissible to C determined?* The rationale that we use to decide that are far from arbitrary, but motivated by the laws that govern the domain in reality. In [18], the philosopher of science Mario Bunge defines the concepts of a *state space* of a thing<sup>2</sup>, and a subset of it, which he names a *nomological state space* of a thing. The idea is that among all the (theoretically) possible states a thing can assume, only a subset of it is lawful and, thus, is actually possible. Additionally, he defends that the only really possible *facts* involving a thing are those that abide by laws, i.e., those delimited by the nomological state space of thing. As a generalization, if an actual state of affairs consists of facts [19], then the set of possible state of affairs is determined by a *domain nomological state space*. In sum, possibility is not by any means defined arbitrarily, but should be constrained by the set of laws that constitute reality. For example, it is law of the domain (in reality) that every Father is a Person. The specification (T<sub>2</sub>) is an ideal ontology for C because it includes the representation of this law of this domain via the subsumption relation between the corresponding representations of father and person. Conversely, if C included a world structure in which this law would be broken, the conceptualization itself would not be truthful to reality. To refer once more to Ullmann’s triangle (Figure 1), the relation between a conceptualization C and the *domain nomological state space* is that relation of *abstracts* between conceptualization and reality.

Now, to raise the level of abstraction, we can also consider the existence of a meta-conceptualization C<sub>0</sub>, which defines the set of all domain conceptualizations such as C

<sup>2</sup> The word Thing is used by Bunge in a technical sense, which is synonymous to the notion of *substantial individual* as used in [10].



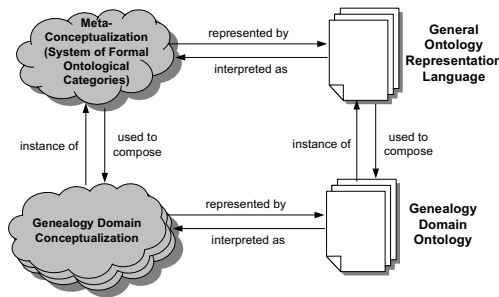
that are truthful to reality. Our main objective is to define a general ontology representation language  $\mathcal{L}_0$  that can be used to produce domain ontologies such as  $\mathcal{O}_C$ , i.e., a language whose primitives include theories that help in the formal characterization of a domain-specific language  $\mathcal{L}$ , restricting its logical models to those deemed admissible by  $\mathcal{C}$ . In order to do this, we have to include primitives in language  $\mathcal{L}_0$  that represent the laws that are used to define the nomological world space of meta-conceptualization  $\mathcal{C}_0$ . In this case, these are the general laws that describe reality, and describing these laws is the very business of *formal ontology* in philosophy.

In summary, we defend that the ontology underlying a general ontology representation language  $\mathcal{L}_0$  should be a meta-ontology that describes a set of real-world categories that can be used to talk about reality, i.e., ontology in the sense (D2) of introduction. Likewise, the axiomatization of this meta-ontology must represent the laws that define that nomological world space of reality. This meta-ontology, when constructed using the theories developed by *formal ontology* in philosophy, is named a *foundational ontology*.

We can summarize the main points of this latter discussion as follows. A domain such as genealogy is what is named in the literature a *material domain* [20] and a language designed to model phenomena in this domain is called a *domain-specific language*. According to the language evaluation framework mentioned in Section 1, we can provide the following definition for an ideal language to represent phenomena in a given domain:

*A language ideal to represent phenomena in a given domain if the metamodel of this language is isomorphic to the ideal ontology of that domain, and the language only has as valid specifications those whose logical models are exactly the logical models of the ideal ontology.*

This principle should hold not only for domain-specific languages, whose metamodels should be isomorphic to some ontology of a material domain, but also for domain-independent languages and, in particular, for general ontology representation languages that can be used to create domain ontologies in different material domains. To be consistent with the position defended here, a language  $\mathcal{L}_0$  used to represent individual domain ontologies should also be based on a conceptualization, but in this case, a meta-conceptualization, which is represented by a *Foundational Ontology*. This idea is illustrated in Figure 7.



**Figure 7.** Relations between Material Domain Conceptualization, Domain Ontologies, General Ontology Representations Languages and their Meta-conceptualization as a formal system of ontological categories.

#### 4. Towards a suitable General Ontology Representation Language: *The Ontological Level Revisited*

When a general ontology representation language is constrained in such a way that its intended models are made explicit, it can be classified as belonging to the *ontological level*. This notion has been proposed by Nicola Guarino in [21], in which he revisits Brachman's classification of knowledge representation formalisms [22].

In Brachman's original proposal, the modeling primitives offered by knowledge representation formalisms are classified in four different levels, namely: *implementation*, *logical*, *conceptual* and *linguistic* levels.

In the logical level, we are concerned with the predicates necessary to represent the concepts of a domain and with evaluating the truth of these predicates for certain individuals. The basic primitives are propositions, predicates, functions and logical operators, which are extremely general and ontologically neutral. For instance, suppose we want to state that a red apple exists. In predicate calculus we would write down a logical formula such as  $(F_1) \exists x(\text{apple}(x) \wedge \text{red}(x))$ . Although this formula has a precise semantics, the real-world interpretation of a predicate occurring in it is completely arbitrary, since one could use it to represent a property of a thing, the kind the thing belongs to, a role played by the thing, among other possibilities. In this example, the predicates *apple* and *red* are put in the same logical footing, regardless of the nature of the concept they represent and the importance of this concept for the qualification of predicated individual. Logical level languages are neutral w.r.t. ontological commitments and it is exactly this neutrality that makes logic interesting to be used in the development of scientific theories. However, it should be used with care and not directly in the development of ontologies, since one can write perfectly correct logical formulas, but which are devoid of ontological interpretation. For example, the entailment relation has no ontic correlation. Moreover, while one can negate a predicate or construct a formula by a disjunction of two predicates, in reality, there are neither negative nor alternative entities [18].

In order to improve the "flatness" of logical languages, Brachman proposes the introduction of an *epistemological level* on top of it, i.e., between the logical and conceptual levels in the original classification. Epistemology is the branch of philosophy that studies "*the nature and sources of knowledge*". The interpretation taken by Brachman and many other of the logicist tradition in AI is that knowledge consists of propositions, whose formal structure is the source of new knowledge. Examples of representation languages belonging to this level include Brachman's own KL-ONE [23] and its derivatives (including the semantic web languages OIL, DAML, DAML+OIL, RDFS, OWL) as well as object-based and frame-based modeling languages such as EER, LINGO [24] and UML. The rationale behind the design of epistemological languages is the following: (i) the languages should be designed to capture interrelations between pieces of knowledge that cannot be smoothly captured in logical languages; (ii) they should offer *structuring* mechanisms that facilitate understanding and maintenance, they should also allow for economy in representation, and have a greater computational efficiency than their logical counterparts; (iii) finally, modeling primitives in these languages should represent structural connections in our knowledge needed to justify conceptual inferences in a way that is independent of the meaning of the concepts themselves.

Indeed languages such as UML, ER, LINGO and OWL offer powerful structuring mechanisms such as classes, relationships (attributes) and subclassing relations.

However, if we want to impose a certain *structure* in the representation of formula ( $F_1$ ), in a language such as UML, we would have to face the following structuring choices: (a) consider that there are instances of apples that can possess the property of being red or, (b) consider that there are instances of red things that can have the property of being apples. Formally we can state either that ( $F_2$ )  $\exists x: \text{Apple}.\text{red}(x)$  as well as ( $F_3$ )  $\exists x: \text{Red}.\text{apple}(x)$ , and both these many-sorted logic formalizations are equivalent to the previous one-sorted axiom. However, each one contains an implicit structuring choice for the sort of the things we are talking about.

The design of epistemological languages puts a strong emphasis on the inferential process, and the study of knowledge is limited to its form, i.e., it is "*independent of the meaning of the concepts themselves*". Therefore, the focus of these languages is more on formal reasoning than on (formal) representation. Returning to our example, although the representation choice (b) seems to be intuitively odd, there is nothing in the semantics of a UML class or an OWL concept that prohibits any unary predicate such as red or tall to be modeled as such. In other words, since in epistemological languages the semantics of the primitive "sort" is the same as its corresponding unary predicate, the choice of which predicates correspond to sorts is completely left to the user.

In [21], Guarino points out that structuring decisions, such as this one, should not result from heuristic considerations but instead should be motivated and explained in the basis of suitable *ontological distinctions*. For instance, in this case, the choice of Apple as the sort (a) can be justified by the meta-properties that we are ascribed to the term by the *intended meaning* that we give to it. The ontological difference between the two predicates is that Apple corresponds to a Natural *Kind* whereas Red corresponds to an Attribution or a *Mixin* [10]. Whilst the former applies necessarily to its instances (an apple cannot cease to be an apple without ceasing to exist), the latter *only* applies contingently. Moreover, whilst the former supplies a *principle of identity*<sup>3</sup> for its instances, i.e., a principle through which we judge if two apples are numerically the same, the latter cannot supply one. However, it is not the case that an object could subexist without obeying a principle of identity [25], an idea which is defended both in philosophical ontology (e.g., Quine's dicto "*no entity without identity*" [14]), and in conceptual modeling (e.g., Chen's design rational for ER [15]). Consequently, the structuring choice expressed in ( $F_3$ ) cannot be justified.

In addition to supporting the justified choice for structuring decisions, the ontological level has important practical implications from a computational point of view. For instance, one can exploit the knowledge of which predicates hold necessarily (and which are susceptible to change) in the design and implementation of more efficient update mechanisms. Finally, there are senses in which the term Red can be said to hold necessarily (e.g., "*scarlet is a type of red*" referring to a particular shade of color), and senses in which it carries a principle of identity to its instances (e.g., "*John is a red*" - meaning that "*John is a communist*"). The choice of representing Red as a *Mixin* in the aforementioned representation makes explicit the intended meaning of this predicate, ruling out these two other possible interpretations. In epistemological and logical languages, conversely, the intended meaning of a predicate relies on our understanding of the natural language label used. As this example makes explicit, an ontologically well-founded modeling language should commit to a system of

<sup>3</sup> For an extensive discussion on *kinds*, *attributions* and *principles of identity* as well as their importance for the practice of conceptual modeling we refer to [10].

ontological meta-level categories that include, for instance, *Kinds*, *Roles*, and *Mixin* as distinctions which further qualify and make precise and explicit the real-world semantics of the terms used in domain representations. An example a foundational ontology that comprises such a system of categories is discussed in the next section.

## 5. Domain-Independent and Domain-Specific Languages: an Illustrative Example

The language evaluation and design framework briefly discussed in Section 2 can be applied both at the level of material domains (e.g., genomics, archeology, multimedia, fishery, law, etc.) and corresponding domain-specific modeling languages, and at the (meta) level of a domain-independent (meta) conceptualization that underpins a general conceptual (ontology) modeling language. In [10], we have developed a Foundational Ontology named **UFO (Unified Foundational Ontology)** which can be used as theoretically sound basis for evaluating and redesigning conceptual modeling languages, in general, and ontology representation languages in particular. In Figure 8, we illustrate a small excerpt of this foundational ontology that contains a typology of universals (roughly classes, types).

In [10], we used this fragment of the UFO ontology to evaluate and re-design the portion of UML dealing with classifiers for the purpose of conceptual modeling and ontology representation. The re-designed UML 2.0 metamodel resulting from this process is depicted in Figure 9. This metamodel describes the abstract syntax of (part of) a general ontology representation language. The UML profile implementing this

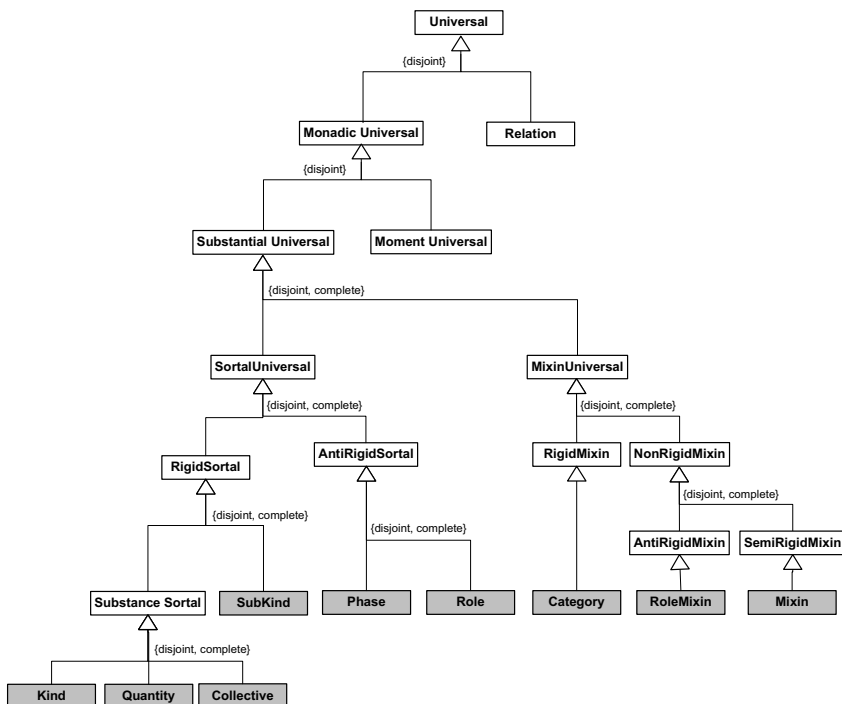


Figure 8. Excerpt of the Foundational Ontology UFO depicting a typology of universals [10].

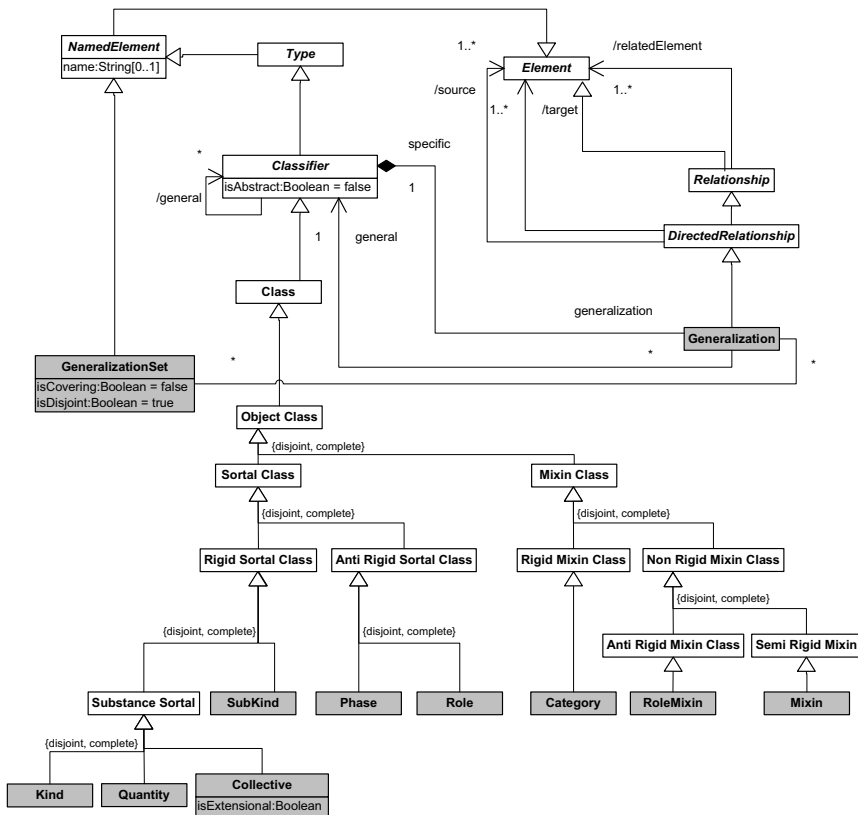


Figure 9. Redesigned UML 2.0 metamodel according to the Foundational Ontology of Figure 8.

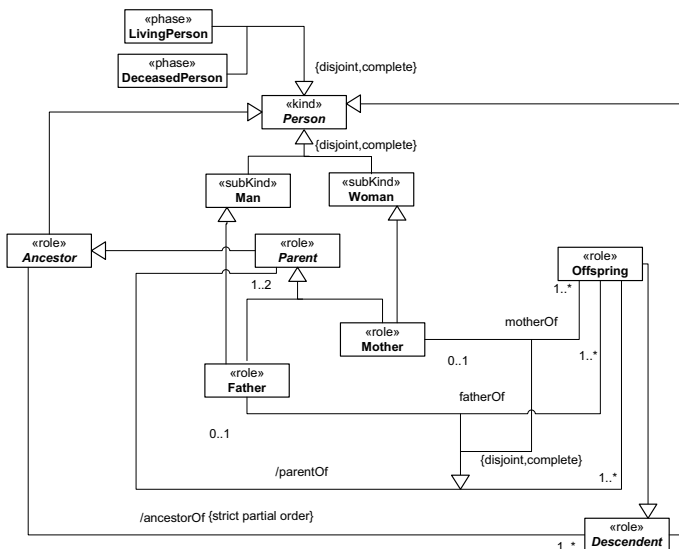


Figure 10. An ontology for the genealogy domain.











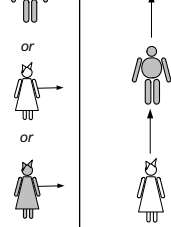
Language					 <i>(is-arrow-directly-connected)</i>	 or 	 or  or 	 <i>composition of is-arrow-path-connected with the above relation in the plane, e.g.</i>	
Ontology	Living Man	Deceased Man	Living Woman	Deceased Woman	Parent of	Father	Mother	Offspring	Ancestor of

Figure 11. Domain Concepts and their representing modeling primitives in language L<sub>1</sub>.

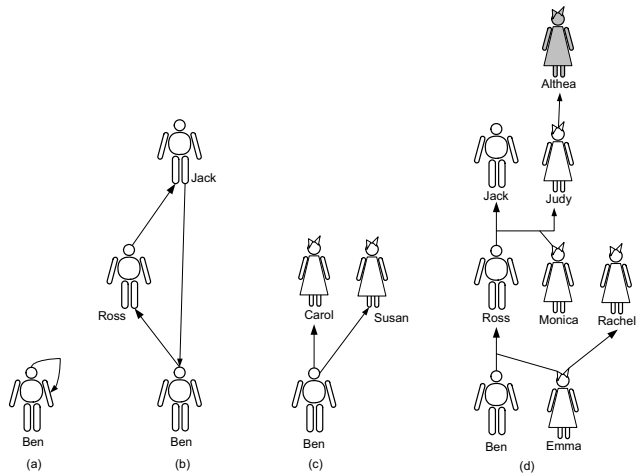
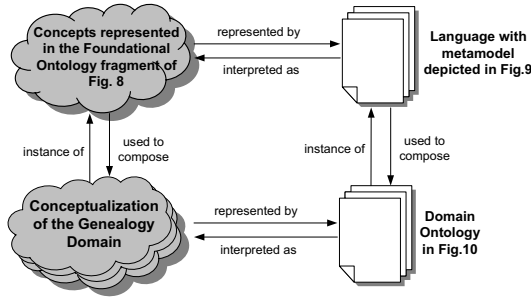


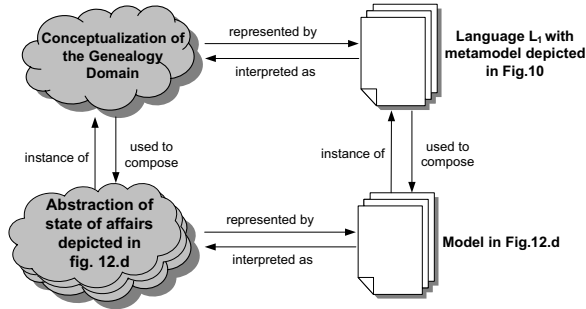
Figure 12. (a-c) Examples of invalid models and (d) of a valid model in L<sub>1</sub>.

metamodel is illustrated in Figure 10, in which it is used to represent an ontology for the genealogy material domain. In [10], we also used this domain ontology and the framework discussed in Section 1 to systematically design a domain-specific modeling language in the domain of Genealogy (named hereafter L<sub>1</sub>). The modeling primitives of L<sub>1</sub> are depicted in Figure 11. Figure 12 presents examples of invalid (Figure 12a-c) and valid models (Figure 12d) produced using that language.

By instantiating the pattern of Figure 2 to the whole example discussed so far we obtain the correspondences depicted in Figures 13 and 14. The ontology O of Figure 10 is a concrete representation of a given conceptualization of the genealogy domain. In this case, we have the ideal situation that the metamodel of language L<sub>1</sub> is isomorphic to this ontology. The genealogy concepts represented in O are used to articulate models of individual state of affairs in reality. A specification in language L<sub>1</sub> (such as the one of Figure 12d) is a concrete artifact representing one of these models. Since L<sub>1</sub> is an



**Figure 13.** Instantiating the pattern of Figure 2 for a domain-independent meta-conceptualization.



**Figure 14.** Instantiating the pattern of Figure 2 for the domain of genealogy.

ideal language to represent the genealogy domain according to the ontology of Figure 10, the only grammatically valid models of this language are the ones which represent abstractions which are deemed acceptable according to that ontology. For this reason, models such as the ones that represent abstractions in which someone is his own father/ancestor (12a), or a father/ancestor of one of his ancestors (12b) are invalid models in this language. Finally, a domain ontology such as the one just discussed is also a concrete artefact (a model), and as much as the models in Figure 12, it must be represented in some modeling language. An example of such a language is the version of UML (the UML profile) used in Figure 10. This modelling language has a metamodel (Figure 9) which is isomorphic to the foundational ontology whose fragment is depicted in Figure 8. Here once more, the grammatically valid models (domain ontologies) according to this UML profile are only those that represent (domain) conceptualizations which are deemed accepted by the UFO ontology. So, for instance, one cannot produce in this language a conceptualization in which (i) *a Role is supertype of a Kind*, or that (ii) *an Object Class is not a subkind of exactly one Kind*<sup>4</sup>.

## 6. Reference and Lightweight Ontologies

Conceptual Models vary in the way they manage to represent an associated conceptualization. An ontology such as the one in Figure 10 is more accurate than if it

<sup>4</sup> These two constraints (i) and (ii) have been formally proved in [10].

were represented in ER, OWL, LINGO or standard UML. This is because the modeling profile used in that model commits to a much richer meta-ontology than the ones underlying these other languages. As a consequence, to formally characterize its ontological distinctions, a formal language with higher expressiveness is needed. When the stereotyped modeling primitives of this profile are used, an axiomatization in the language of *intensional modal logics* is incorporated in the resulting specification, constraining the interpretation of its terms<sup>5</sup>. Quantified Intensional modal logics are more expressive than, for example, a *SHOIN*( $D_n$ ) descriptions logics, which is the language behind the formalization of OWL. In contrast, a language such as OWL has been carefully designed to maintain interesting properties such as computational tractability and decidability, which are properties that are in general absent in more expressive languages. Likewise, LINGO was designed to facilitate the translation to

Object-Oriented implementations. Properties such as computational efficiency and easiness of translation to implementation platforms have been recognized as important properties to application areas of ontology in computer science such as the Semantic Web initiative and Domain Engineering [24]. Therefore, in Ontology Engineering, the following tradeoff must be recognized. On one side we need a language that commits to a rich foundational ontology. This meta-ontology, however, will require the use of highly expressive formal languages for its characterization, which in general, are not interesting from a computational point of view. On the other side, languages that are efficient computationally, in general, do commit to a suitable meta-conceptualization. The obvious question is then: how can we design a suitable general ontology representation language according to these conflicting requirements?

The position advocated here is analogous to the one defended in [26], namely, that we actually need two classes of languages. We explain this position by once more making use of an analogy to the engineering processes in the disciplines of Software and Information Systems Engineering. In both disciplines, there is a clear distinction between Conceptual Modeling, Design and Implementation. In Conceptual Modeling, a solution-independent specification is produced whose aim is to make a clear and precise description of the domain elements for the purposes of communication, learning and problem-solving. In the Design phase, this conceptual specification is transformed in a design specification by taking into consideration a number of issues ranging from architectural styles, non-functional quality criteria to be maximized, target implementation environment, etc. The same conceptual specification can potentially be used to produce a number of (even radically) different designs. Finally, in the Implementation phase, a design is coded in a target language to be then deployed in a computational environment. Again, from the same design, a number of different implementations can be produced. Design, thus, bridges Conceptual Modeling and Implementation.

We here defend an analogous principle for Ontology Engineering. On one hand, in a conceptual modeling phase in Ontology Engineering, highly-expressive languages should be used to create strongly axiomatized ontologies that approximate as well as possible to the ideal ontology of the domain. The focus on these languages is on representation adequacy, since the resulting specifications are intended to be used by humans in tasks such as communication, domain analysis and problem-solving. The resulting domain ontologies, named *reference ontologies* in [16], should be used in an

---

<sup>5</sup> The complete formal semantics of this profile in a system of Modal Logics with Sortal Quantification is presented in [10].



*off-line* manner to assist humans in tasks such as meaning negotiation and consensus establishment. On the other hand, once users have already agreed on a common conceptualization, versions of a reference ontology can be created. These versions have been named in the literature *lightweight ontologies*. Contrary to reference ontologies, lightweight ontologies are not focused on representation adequacy but are designed with the focus on guaranteeing desirable computational properties. Examples of languages suitable for lightweight ontologies include OWL and LINGO. An example of an ontology representation language that is suitable for reference ontologies is the UML profile briefly illustrated in the previous section (as demonstrated in [10]). It is important, nonetheless, to highlight that, as discussed in the previous section, languages such as OWL and LINGO are epistemological level languages and, thus, naming them ontology representation languages is actually a misnomer. Finally, a phase is necessary to bridge the gap between the conceptual modeling of reference ontologies and the coding of these ontologies in terms of specific lightweight ontology languages. Issues that should be addressed in such a phase are, for instance, determining how to deal with the difference in expressivity of the languages that should be used in each of these phases, or how to produce lightweight specifications that maximize specific non-functional requirements (e.g., evolvability vs. reasoning performance).

Finally, the importance of reference ontologies has been acknowledged in many cases in practice. For instance, [27] illustrates examples of semantic interoperability problems that can pass undetected when interoperating lightweight ontologies. Likewise, [28] discusses how a principled foundational ontology can be used to spot inconsistencies and provide solutions for problems in lightweight biomedical ontologies. As a final example, the need for methodological support in establishing precise meaning agreements is recognized in the *Harvard Business Review* report of October 2001, which claims that “*one of the main reasons that so many online market makers have foundered [is that] the transactions they had viewed as simple and routine actually involved many subtle distinctions in terminology and meaning*”.

## 7. Summary

In the sequel, we summarize the most important points defended in this article:

1. *Formal Ontology*, as conceived by Husserl, is part of the discipline of Ontology in philosophy (sense D1), which is, in turn, the most important branch of metaphysics.
2. Formal Ontology aims at developing general theories that accounts for aspects of reality that are not specific to any field of science, be it physics or conceptual modeling (sense D2).
3. These theories describe knowledge about reality in a way, which is independent of language, of particular states of affairs (states of the world), and of epistemic states of knowledgeable agents. In this article, these language independent theories are named (meta) *conceptualizations*. The representation of these theories in a concrete artifact is a *foundational ontology*.
4. A Foundational Ontology is domain-independent Reference Ontology. Reference Ontologies try to characterize as accurately as possible the conceptualization they commits to.
5. Foundational ontologies in the philosophical sense can be used to provide real-world semantics for general *ontology representation languages* and to constrain the possible interpretations of their modeling primitives. Conversely, a suitable

ontology representation language should commit to a system of ontological distinctions (in the philosophical sense), i.e., they should truly belong to the *Ontological Level*.

6. An ontology can be seen as the *metamodel* specification for an ideal language to represent phenomena in a given domain in reality, i.e., a language which only admits specifications representing possible state of affairs in reality (related to sense D3).
7. Suitable general conceptual modeling languages can be used in the development of reference *domain ontologies*, which, in turn, among many other purposes, can be used to characterize the vocabulary of *domain-specific languages*.
8. The discipline of Ontology Engineering should account for two classes of languages with different purposes: (i) on one hand, it needs well-founded ontology representation languages focused on representation adequacy regardless of the consequent computational costs, which is not actually a problem since the resulting model is targeted at human users; (ii) On the other hand, it needs lightweight representation languages with adequate computational properties to guarantee their use as codification alternatives for the reference ontologies produced in (i).
9. The name ontology representation languages when applied to the so-called Semantic Web languages is a misnomer, since these languages are motivated by epistemological and computational concerns, not ontological ones.

## Acknowledgements

The author would like to thank Gerd Wagner, Nicola Guarino, Luís Ferreira Pires, Marten van Sinderen, Renata S.S. Guizzardi and Chris Vissers for fruitful discussions and for providing valuable input to the issues of this article.

## References

- [1] Merriam-Webster Dictionary, [online: [www.m-w.com](http://www.m-w.com)], captured in 2004.
- [2] Mealy, G. H. 'Another Look at Data', Proc. of the Fall Joint Computer Conference, Anaheim, California (AFIPS Conference Proceedings, Volume 31), Washington, DC: Thompson Books, London: Academic Press, 525–534, 1967.
- [3] Quine, W. V. O. 'On What There Is', as reprinted in *From a Logical Point of View*, New York: Harper & Row, 1953.
- [4] Hayes P. 'The Naive Physics Manifesto', In D. Ritchie (Ed.) *Expert Systems in Microelectronics age*. Edinburgh University Press, pp. 242-270, 1978.
- [5] Hayes, P. 'Naive Physics I: Ontology for Liquids', in Hobbs & Moore, pp. 71-107, 1985.
- [6] Ullmann, S. 'Semantics: An Introduction to the Science of Meaning', Basil Blackwell, Oxford, 1972.
- [7] Ogden, C. K., Richards, I. A. 'The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism', New York: Harcourt Brace Jovanovich, 1923.
- [8] de Saussure, F. 'Course in General Linguistics', Roy Harris (trans.), Open Court Publishing Company, 1986 (original from 1916).
- [9] Baldinger, K. 'Semantic Theory: Towards a Modern Semantics', Palgrave Macmillan, 1980.
- [10] Guizzardi, G. "Ontological Foundations for Structural Conceptual Models", *Telematica Instituut Fundamental Research Series no. 15*, Universal Press, The Netherlands, 2005, ISBN 90-75176-81-3.
- [11] Guizzardi, R.S.S.; Guizzardi, G. "Integrating Agent-Oriented Modeling Languages Using a Foundational Ontology", in *Social Modeling for Requirements Engineering*, P. Giorgini, N. Maiden, J. Mylopoulos, E. Yu (eds.), *Cooperative Information Systems Series*, MIT Press, 2007 (to appear).
- [12] Grice, H.P. 'Logic and conversation', In: *Syntax and Semantics: Vol 3, Speech Acts* (P. Cole & J. Morgan, eds). Academic Press, New York, pp. 43-58, 1975.

- [13] Milton, S.K., Kamierczak, E. 'An Ontology of Data Modeling Languages: A Study Using a Common-Sense Realistic Ontology', *Journal of Database Management, Special Issue on Ontological Analysis, Evaluation, and Engineering Of Business Systems Analysis Methods*, 18 pages, 2004.
- [14] Quine, W. V. O. 'Ontological relativity', and Other Essays, New York: Columbia University Press, 1969.
- [15] Chen, P. 'The entity-relationship model: Towards a unified view of data', *ACM Transactions on Database Systems* 1(1), 1976.
- [16] Guarino, N. 'Formal Ontology and Information Systems', In Guarino, N. (ed.) *Formal Ontology in Information Systems, Proceedings of the International Conference on Formal Ontology and Information Systems (FOIS)*, Trento, Italy, June 6-8. IOS Press, Amsterdam: pp. 3-15, 1998.
- [17] Ciocoiu, M., Nau D. 'Ontology-Based Semantics', In: *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, Breckenbridge, Colorado, April 12-17, 2000.
- [18] Bunge M. 'Ontology I: The Furniture of the World', *Treatise on Basic Philosophy*, vol. 3, D. Reidel Publishing, New York, 1977.
- [19] Armstrong, D.M. 'A World of State of Affairs', *Cambridge Studies in Philosophy*, Cambridge University Press, 1997.
- [20] Smith, B. 'Logic and Formal Ontology', in J. N. Mohanty and W. McKenna, eds., *Husserl's Phenomenology: A Textbook*, Lanham: University Press of America, 29-67, 1989.
- [21] Guarino, N. 'The Ontological Level', In R. Casati, B. Smith and G. White (eds.), *Philosophy and the Cognitive Science*. Holder-Pivhler-Tempsky, Vienna: pp. 443-456, 1994.
- [22] Brachman, R. J. 'On the Epistemological Status of Semantic Networks', In N. V. Findler (Ed.), *Associative Networks: Representation and Use of Knowledge by Computers*. Academic Press, 1979.
- [23] Brachman, R. J. and J. G. Schmolze 'An Overview of the KL-ONE Knowledge Representation System', *Cognitive Science* 9, 171-216. Carrara, M. 1992. *Identità e persona nella riflessione*, 1985.
- [24] Falbo, R. A., Guizzardi, G., Duarte, K. C. 'An Ontological Approach to Domain Engineering', In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Ischia, Italy, 2002.
- [25] van Leeuwen, J. 'Individuals and sortal concepts : an essay in logical descriptive metaphysics', PhD Thesis, University of Amsterdam, 1991.
- [26] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A. 'Ontology Library', WonderWeb Deliverable D18, 2003.
- [27] Guizzardi, G. "The Role of Foundational Ontology for Conceptual Modeling and Domain Ontology Representation", Keynote Paper, 7th International Baltic Conference on Databases and Information Systems, Vilnius, Lithuania, 2006.
- [28] Fielding, J.M.; Simon, J.; Ceusters, W.; Smith, B. 'Ontological Theory for Ontology Engineering'. in: *Proceedings of KR 2004, Ninth International Conference on the Principles of Knowledge Representation and Reasoning*, Whistler, Canada, 2004.

# Pattern Repositories for Software Engineering Education

Hans-Werner SEHRING, Sebastian BOSSUNG, Patrick HUPE,  
Michael SKUSA and Joachim W. SCHMIDT

{hw.sehring,sebastian.bossung,pa.hupe,skusa,j.w.schmidt}@tuhh.de  
Software Systems Institute (STS)  
Hamburg University of Science and Technology (TUHH)

**Abstract.** Modern software engineering attacks its complexity problems by applying well-understood development principles. In particular, the systematic adoption of *design patterns* caused a significant improvement of software engineering and is one of the most effective remedies for what was formerly called the *software crises*. Design patterns and their utilization constitute an increasing body of knowledge in software engineering. Due to their regular structure, their orthogonal applicability and the availability of meaningful examples design patterns can serve as an excellent set of use cases for organizational memories, for software development tools and for e-learning environments.

Patterns are defined and described on two levels [1]: by real-world examples—e.g., textual or graphical content on their principles, best practices, structure diagrams, code etc.—and by conceptual models—e.g., on categories of application problems, software solutions, deployment consequences etc. This intrinsically dualistic nature of patterns makes them good candidates for conceptual content management (CCM). In this paper we report on the application of the CCM approach to a repository for teaching and training in pattern-based software design as well as for the support of the corresponding e-learning processes.

**Keywords.** Conceptual modeling, content management, design patterns, e-learning

## Introduction and Motivation

The entire field of modern software engineering (SE) is a diverse and complex endeavor. Many advances had to be made to master what we used to call the software crisis. One important key to success was finally found in software patterns which introduced another level of abstraction to software design processes and decreased the complexity of designs considerably. Since their first publication in the early 90ies [1] design patterns (and the idea of software patterns in general) have been quickly adopted and today are in widespread use.

As modern software systems become ever larger and more complex most software development paradigms agree that some planning-ahead is needed in order to successfully carry out a software development project [2]. While requirements for the application under development are collected during an initial application analysis phase, the subsequent design phase aims at a coarse design of the software. In this light, design

patterns are abstractions over pieces of software with shared design requirements thus helping to exploit accumulated design experience.

Design patterns turn out to be essential pieces of knowledge for software engineers and, consequently, have to be covered by software engineering curricula. As the mere awareness of a design pattern does not enable a student of software engineering to appropriately apply it, the pragmatics of pattern applications should also be taught. This can be achieved by providing pattern definitions together with best-practice pattern applications and by enabling students to relate, publicize and discuss their personal pattern applications in such definitional contexts. Students can thus actively participate in the design process, resulting in an improved learning experience.

In previous projects we have used our Conceptual Content Management (CCM) approach to provide extensional concept definitions and to embed such CCM material into learning environments—albeit in political iconography [3] or art history applications [4]. In this paper we sketch a conceptual model for design patterns and show how previous experiences with active learning can be carried over to SE. We discuss how existing work on patterns (e.g., [1]) can be represented, communicated, and applied by CCM.

The remainder of this paper is organized as follows: We commence with an overview of requirements to SE tools and their relationships with content management in Section 1. In Section 2 we provide a conceptual model suitable for describing patterns in learning systems and also report on some related work. Section 3 describes pattern learning scenarios with particular emphasis on active learning in conceptual content management systems. We conclude with a summary and outlook in Section 4.

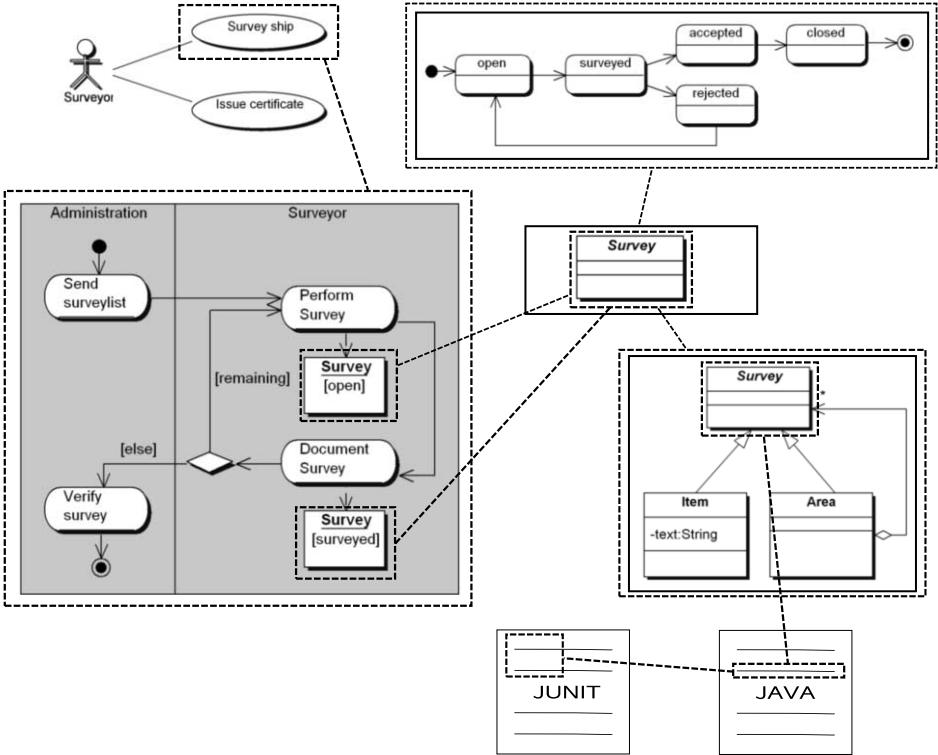
## 1. Requirements for Software Engineering Tools

As a basis for discussion this section contains some remarks on SE processes, in particular on the pattern-based design of software, and we argue that SE—especially the aspect of conserving design knowledge—shares properties of advanced content management tasks.

### 1.1. *Software Engineering Processes*

A typical SE process consists of several phases: analysis, design, implementation and test (e.g., [2]). During these phases, numerous artifacts are created. These artifacts stem from different models and coexist for various reasons: different levels of abstraction, perspectives, revisions, or alternate representations.

Typically, artifacts of later phases are created from higher-level artifacts of earlier phases, but dependencies are not linear and difficult to trace: Diverse analysis requirements, platform constraints and modeling acts such as architectural decisions, pattern applications, etc. manifest themselves in later artifacts. While the knowledge about these manifestations is available at the point of decision-making, this information often is not part of an SE process [5]. Instead, such information has to be recorded in additional documentation which accompanies the process. Only recently efforts have begun to interconnect these artifacts at least partially [6]. The conservation and communication of development experience is an issue not systematically addressed by SE. Figure 1 shows a number of typical software engineering artifacts from one of our projects. Superimposed



**Figure 1.** Examples of interrelated software engineering artifacts

onto these are the interrelationships between them (dashed lines), which are neither explicit in the artifacts themselves nor handled by the corresponding tools in a coherent and global manner.

1.2. Design Phase Activities

One of the phases of typical SE processes is that of design. This phase gains much attention because it is the point where reuse of development experience can take place at a high level.

The application of design patterns has become an important member of the set of design activities (Figure 2). The reason for this is that such applications are well-understood cases of software design. Design patterns also constitute a growing body of knowledge and best-practices. They can be used to preserve and communicate design experience.

These reasons—design patterns being well-understood and being a medium to communicate development experience—makes them a germane means for teaching software development. We apply patterns in SE education as a first object of study towards a model-based treatment of design patterns. As a first step in this direction we treat specific patterns (see Figure 2).

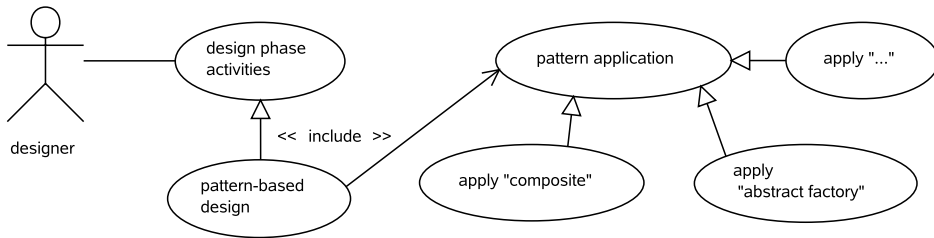


Figure 2. Design use cases

### 1.3. Software Engineering as a Content Management Activity

We have studied how entities are represented by means of content management with regard to SE [7]. Emphasis is put on the representation of entities that cannot fully be represented by data records alone, especially items that have subjective interpretations. We refer to the research of such content management applications as *conceptual content management* (CCM).

Based on the epistemic observation that neither content nor concepts exist in isolation, CCM is based on the conjoint description of entities by pairs of content representations and conceptual models. For pairs of these two we introduce the notion of *assets*. *Content* is presented by arbitrary multimedia documents. *Concepts* consist of the *characteristic* properties of entities, *relationships* with other assets which describe the interdependencies of entities, and *constraints* on those characteristics and relationships.

The statements of the asset language allow the definition of asset classes, the creation and manipulation of asset instances, and their retrieval. Some examples follow. For a more complete description of the asset language see [8].

The following sketches asset definitions to model an SE processes:

```

model SoftwareEngineering
class SoftwareModel {
  content xmiDocument :org.w3c.dom.Document
  concept relationship classes :ClassDescription*
  relationship sequences :Sequence*
  constraint operationsDefined
    sequences.objActs.msgs <= classes.operations
    and ... ; matching signatures
  :
}
class Sequence {
  content topNode :org.w3c.dom.Element
  concept relationship objActs :ObjectActivation*
}
class ObjectActivation ...
  
```

In the example, an XML document is the content of a general `SoftwareModel`, presumably an XMI representation of UML diagrams. Related instances of further asset classes are used to describe parts of the software model—classes, sequences, etc.—in

more detail. Such an asset model can be created by the structural conversion of an existing model for software in general, for example the Meta Object Facility (MOF, see [9]).

SE entities are described by the aforementioned characteristic attributes and relationships. Possible types of content handles and characteristics are determined by an embedded language which also is the target language of the model compiler (currently Java). More extensive use of the asset model is achieved by employing constraints that reflect the rules of the chosen SE process. In the example this is demonstrated by a constraint `operationsDefined` which in addition to the semantics of the UML ensures that for each message sent according to a sequence diagram a method with a matching signature is defined. When this constraint is violated, the corresponding instances can either not be allowed at all, or—in case of an interactive system—the situation can be brought to the attention of the user, e.g., on a list of issues.

Since assets are especially designed to support subjective views, asset classes as well as instances can be *personalized* by means of redefinitions on an individual basis. Based on the above definition of `SoftwareModel` a user can define

```

model MySoftwareEngineering
from SoftwareEngineering import SoftwareModel
class SoftwareModel {
    concept relationship objects :ObjectDescription*
    constraint objects.type <= classes
}

```

to explicitly refer to instances and to require that all of their classes have to be part of the model. Anything that is not mentioned in the personalization remains the same as in the original definition.

For asset redefinitions there is a demand for *openness* and *dynamics*. We call a *CCM system (CCMS)* open if it allows users to define assets according to their current information needs. Dynamics is the ability of a system to follow redefinitions of assets at runtime without interrupting the users' work.

To account for dynamics, our approach to CCM consists of three main contributions [10]: an *asset language* for the description of entities by both content and conceptual expressions, a *modularized architecture* for evolving conceptual content management systems, and a *model compiler* which translates expressions given in the asset language into CCMSs without developer intervention.

## 2. Modeling Design Patterns

This section first gives a brief overview of design patterns in general and approaches to create conceptual models for them. We then show how to model design patterns in CCM.

### 2.1. Design Patterns

The central idea of design patterns is to capture solutions to recurring problems. In other words, experiences gained by adept programmers are put into a form that is suitable to pass on these experiences to others, thereby eliminating the need for them to relearn the same knowledge “the hard way” by rediscovering them from practice. The exact workings of such human learning mechanisms are currently under research [5]. This



capturing of short-cuts in learning experiences makes design patterns an important part of SE curricula.

Existing work on design patterns (most prominently Gamma et al, [1]) identifies four central elements of a pattern: A name, a context in which it can be applied, the solution it provides, and the consequences that arise from it. However, [1] also acknowledge that there is some subjectivity in design patterns. It is noted that what is and what is not a pattern depends on the individual user's point of view. Other work [11] also points out the common definition of a design pattern being "a solution to a problem in a context" should be extended to also include information about recurrence as well as about teaching, to provide the means to apply the solution to new situations and to notice that an opportunity to do so has arisen in the first place. This is particularly important with respect to teaching patterns, where a definition of the design patterns is not sufficient. We will describe in Section 3.3 how these aspects can be handled by CCMSs.

## *2.2. Pattern Description Languages*

The need for a pattern description language arises in several contexts which can broadly be divided into those that aim to support the task of creating software (e.g., by pattern application) and those that inspect existing software (e.g., by pattern discovery).

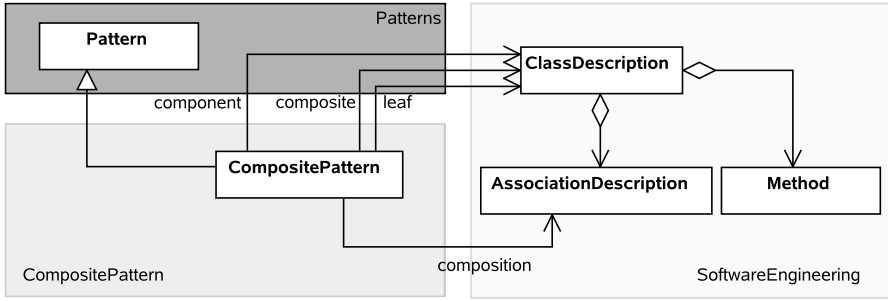
An early metamodel for pattern-based CASE tools is proposed in [12]. Based on this metamodel, pattern instantiation is proposed, relating available patterns to actual application artifacts. UML also offers the collaboration mechanism which can to some extent be used to model design patterns. By itself this is too weak a model for pattern-based tools, which need additional means to capture semantics, e.g., OCL [13]. Yet other tools let programmers work on different levels of abstraction allowing them to work on the source code as well as to instantiate patterns [14].

Metamodels are also employed to extract patterns from existing software. Corresponding tools identify micro-architectures by modeling classes in roles [15], or they describe design pattern applications [16]. The description is geared towards system supported application of the pattern. Taking this one step further, there are approaches to enforce the use of design patterns, e.g., [17].

All these approaches offer metamodels for design patterns with several different foci, but there is a large overlap at their cores. Commonly no clear distinction between pattern description and the general object-oriented metamodel is made. Our metamodel for patterns is loosely similar to most existing models, but aims to improve the separation of general pattern description, object-oriented metamodel and specific pattern descriptions for learning. It thus allows a fine-grained selection of standard models from which personal derivations are used to provide support for active learning scenarios.

## *2.3. A CCM Model for Pattern-based Design*

Design patterns are generally presented in a semi-structured manner. Gamma et al identify four essential parts of a pattern: name, problem, solution and consequences [1]. Further substructuring of these elements is not prescribed, even though most authors try to adopt a uniform heading structure. However, there are also approaches that fully formalize the description of design patterns such as [18]. This can provide well-defined semantics for the descriptions as well as reasoning on them. However, such formalizations are



**Figure 3.** Classes from different models are combined to model patterns. Most of the model is omitted in this figure for conciseness.

hardly useful in teaching patterns as learners need to gain an intuitive understanding to be able to identify situations where the pattern is relevant.

At the root of our conceptual model for patterns is a basic `Pattern` class whose concept offers the four core elements of patterns: name—context, solution and consequences—which are described as content in semi-structured documents.

```

model Patterns
class Pattern {
  content   name       : String
            problem    : StructuredDocument
            solution    : StructuredDocument
            consequences : StructuredDocument

  concept relationship collaborators :ClassDescription
            relationship collaboratorAspects :ClassMember
}

```

Furthermore, a CCMS allows users to model the patterns to any degree of specificity they want. This can even mean that a class is created for one specific pattern alone if this is required by the learning context. We demonstrate this with the composite pattern:

```

class CompositePattern refines Pattern {
  content
    name :String := "Composite"
  concept
    relationship   component  :ClassDescription
    relationship   composite  :ClassDescription
    relationship   leaf       :ClassDescription
    relationship   composition :AssociationDescription
    constraint specialization1 composite.superClass = component
    constraint specialization2 leaf.superClass = component
    constraint aggregation composition.type = composition
      and composition.source.type = composite
      and composition.target.type = base
    :
}

```

The Composite pattern is characterized by properties which are reflected in the asset class `CompositePattern`. There are classes in three roles: `component`, `composite`, and `leaf`. Both `composite` and `leaf` are subclasses of `component` as defined by the constraints `specialization1` and `specialization2`. Instances of the class which fills the `composite` role aggregate instances of the class filling the role `component` (constraint `aggregation`). For this example assume that there is a class `AssociationDescription` for UML associations with at least the three attributes used: `type` which characterizes the kind of association, as well as `source` and `target` which refer to assets describing the roles of the associated objects.

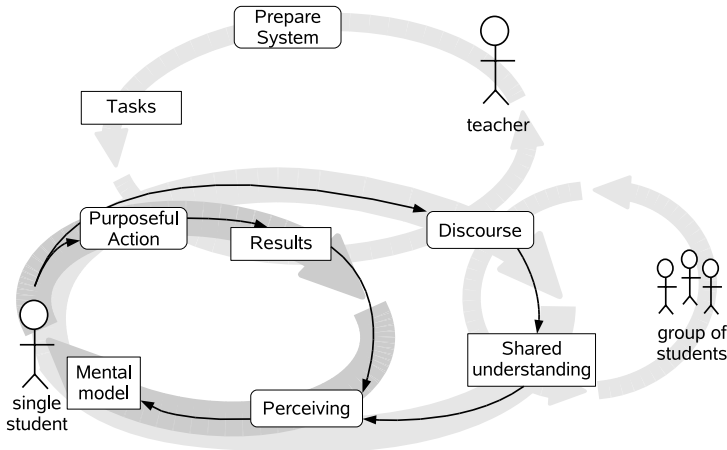
To capture pattern *applications* in a way that is meaningful to students, the pattern applications have to be put into context of the whole software system they were made in. The respective parts of the application domain of the software can be captured by using a `SoftwareEngineering` model (see Section 1.2, [7]). By refining the general pattern model and using the SE model, concrete patterns can be described. Figure 3 gives an overview of the an asset model of the Composite pattern. Instances of this class are used to describe concrete applications of the pattern:

```

model CompositePatternApplication
from SoftwareEngineering import
    ClassDescription, AssociationDescription
from Patterns import CompositePattern
let graphicsPackage := lookfor Package {
    name = "de.tuhh.sts.ltood.figures"
}
let compositeApplication := create CompositePattern {
    problem := ...
    :
    composite :=
        lookfor ClassDescription {
            name = "FigureGroup"
            package = graphicsPackage
        }
    component :=
        lookfor ClassDescription {
            name = "Figure"
            package = graphicsPackage
        }
    leaf :=
        lookfor ClassDescription {
            name="Rectangle"
            package = graphicsPackage
        }
    composition :=
        lookfor Association { source=composite target=component }
}

```

An instance of `CompositePattern` describes a pattern application by providing values for all members. The content members inherited from `Pattern` are filled with



**Figure 4.** Learning cycle from the learner's point of view.

structured documents describing problem, solution, and consequences. These could, e.g., closely correspond to the descriptions typically found in literature on patterns. The conceptual descriptions of the pattern are also provided by retrieving (through the `lookfor` command) appropriate parts of the existing software model. These parts are then explicitly connected as an application of the composite pattern.

### 3. Pattern Learning Scenarios

The application of design patterns has become a commonly accepted design activity. Therefore, teaching the most useful patterns has become an important part of the education or training of software developers. However, applying patterns requires tacit knowledge that cannot be studied on a purely theoretical basis. One needs to learn to actively apply patterns. In this section we argue that active learning is supported well by CCM systems.

A pattern is more than a solution to a problem in a context [11]. Especially for teaching purposes improved descriptions of design patterns are needed. Using the open modeling of the CCM approach such improved descriptions can be formulated, in particular including content. By such combined descriptions, CCM allows active learning processes for design pattern application.

#### 3.1. Active Learning Through Open Dynamic CCM

As is witnessed by many taught courses, understanding content and applying the learnt are essential parts of learning [19]. Constructionists interpret learning as the construction of knowledge and not its absorption [20]. Practical application facilitates the lasting storage of information.

Figure 4 (inspired by [21]) depicts such an active way of learning from the point of view of a student. There are two levels, which differ in closeness to the learner as well as in speed of iteration. The inner cycle constitutes active learning. It is usually carried

out by one learner alone. The outer circle describes the interaction with others, learners as well as teachers through discourse in the field of study and a shared, group-based understanding of it.

Active learning requires—as the name suggests—actions by the user in the field of study. These actions will lead to results, which the learner perceives and understands to be failures or advances. This can then be incorporated into the mental model of the field of discourse and used in the next iteration.

Typical e-learning systems support a passive way of learning: There are usually a number of ways to present lessons to learners [22]. The interaction of learner and system is frequently limited to formal testing.

To take the system support beyond this, the system has to be able to adapt to the particular needs of specific (groups of) learners. We refer to this as personalization, which happens at two levels: *content* and *structure* personalization. The former allows users to adapt the content they work with to their own needs or views of the world. This happens without interfering with the work of other users, but the system needs to provide facilities that allow the later exchange of personalized content between (groups of) users. Structure personalization means that users are not confined to the schema provided for the system, but can modify this schema to suit their needs. This aspect is very important in learning to reflect the level of the learner as well as the field of study. For more details on personalization in e-learning applications see [3].

All activities indicated in the learning circle in Figure 4 are covered by personalization. It supports discourse through exchange of personalized content and conceptual models with a limited group of peer learners. Most importantly, users are enabled to take action in the system itself and learn through the results of their action. They can recombine artifacts to solve learning problems.

This allows an approach to e-learning in which learners can structurally rework or even extend the subject matter. For example, a lesson can start with a given partial model that the learners are to complete. In doing so, they apply what they previously learned. Thinning out the content is again achieved through personalization (the left-out pieces are not deleted globally but hidden in the personal view). A system setup to support this will be shown in Section 3.3.

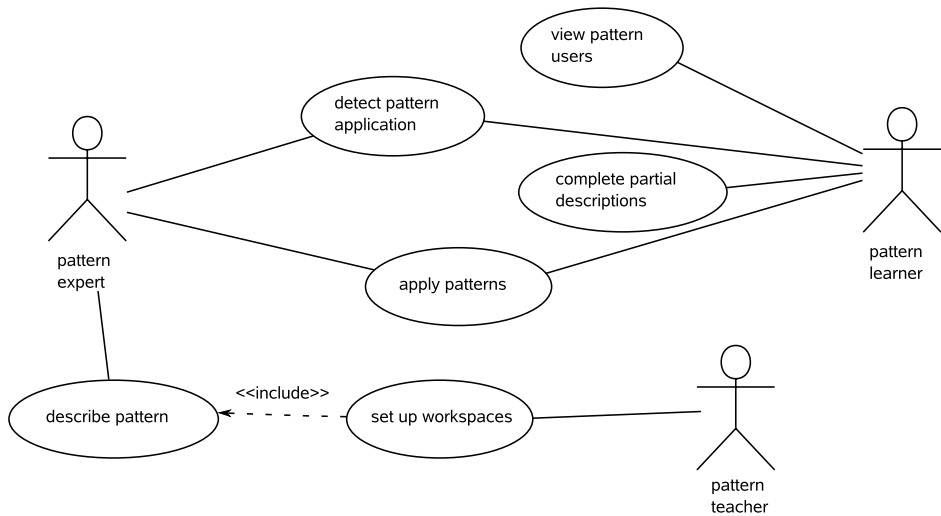
### 3.2. Teaching Pattern-oriented Design

This section applies the CCM approach to learning of design patterns. Particular regard is given to the active dimension of learning.

To use CCM for learning in a field of study, the generic activities of the learning cycle in Figure 4 have to be backed with specific ones of the participants. Figure 5 shows some use cases from which these activities can be deduced. Here a *pattern expert* prepares case studies as examples or master solutions for a *pattern learner*, or the expert gives exercises to be solved by the learner. The person in the role of the *pattern expert* can but does not have to be the teacher at the same time.

In the previous section it has been mentioned that a teacher can hand partial solutions to learners. By means of personalization each learner can solve exercises individually. Personalization furthermore allows change of existing designs to try out modeling alternatives.

Many times, learning begins at the teacher (in the upper cycle in Figure 4) who—in the case of CCM—prepares a system for the particular needs of the learners. This



**Figure 5.** Learning use cases

involves the setup of the general surroundings of the field of study as well as the formulation of particular tasks. When teaching design patterns, the general surroundings are largely described by software engineering in general. The teacher would thus preload the system with a general software engineering model (Section 1.3) which forms a mostly constant basis for the work on design patterns. There is thus little personalization to be expected on the software engineering model, but such personalization is of course possible.

Teaching design patterns requires rich descriptions [11]. In Section 2.3 asset models for typical design pattern descriptions have been shown. Based on the software engineering model, the teacher can import models which describe patterns. Since these are the field of study, heavy personalization is expected here. The openness of the model allows the teacher to extend these models for teaching purposes. Personalizations made by the teacher can lead to concrete tasks for students, for instance where some parts of a model are deleted in the personalization and the task is to fill the created gaps.

Another possible task created through personalization is a scenario in which the teacher provides a description of a concrete software and it is up to the students to detect the use of patterns by creating personal instances of the appropriate pattern applications. Here only class diagrams are given, the learners detect pattern applications by capturing the classes' roles in (personal) Pattern asset instances. More advanced students can be asked to refactor the provided system description through the application of additional patterns where possible.

In both cases it is important to stress that while learners work alone or in small groups they can consult with the other learners of their course or with the teacher as both work in the same system. The system setup that is required to achieve this will be shown in the next section.

System-based learning in general needs support from the system outside the direct field of study: a model of the learners, their advances in the field of study, and a matching with appropriate materials come to mind. Though the above extensions made to the pattern model by the teacher do not automatically lead to a complete model of e-learning

they can be combined with general learning models [23] to form a complete environment of learning.

### 3.3. A CCMS for Teaching Patterns

In order to achieve dynamics, CCMSs are generated from asset models (Section 1.3) by a model compiler [24]. On model changes—a personalization step in the cases considered here—a CCMS is dynamically modified to account for the changed model. Each modification preserves existing asset instances and maintains back references to the public model that has been personalized.

Dynamics of CCMSs is further enabled by an architecture that supports evolution [8]. A CCMS consists of a set of cooperating *components*, each hosting assets of one particular model. Components in turn are implemented by *modules* which offer a certain functionality. The components' functionality is provided by modules working in concert.

Two particular kinds of modules used below are *client modules* and *mediation modules*. Client modules access third-party software such that the services are accessible to a CCMS. A typical example is a client module to map asset definitions to a database for asset persistence.

Mediation modules delegate requests to their two base modules according to a definable strategy which implements a particular behavior. For instance, in a personalization scenario existing asset definitions are stored alongside their personalized variants using two different client modules. A mediation module provides retrieval of assets from both modules, creation of personal assets in the personal client module, and personalization of public assets by copying an asset from a public client module to the personal one and modifying the copy.

Through the generative approach a CCMS for the management of pattern descriptions can be generated from the models presented in Section 2.3. General requirements for such a CCMS can be deduced from [25]. The ability to serve as a tutoring system is based on the consideration of both content and conceptual models in assets and on the modeling openness.

Hosted content describing software systems can serve as an extensional definition of a design pattern by giving an abstract definition and by showing several applications, counter examples, etc. Conceptual models point out the design patterns that are visible in content, for example, in a complete class diagram used as a case study. The dynamic nature of CCMSs permits active learning processes as discussed in the previous sections. For example, students can improve given designs by introducing patterns. In doing they understand how the respective pattern is applied in practice. Furthermore, personalization allows to change definitions of patterns. This way students can experience which properties patterns have and why they are required.

A design pattern CCMS can be set up as a compound system which includes the CCMS generated for an SE model like the one sketched in Section 1.3 as a component. Figure 6 shows an example of such a CCMS generated as a tutoring system. This component can be used independently in SE processes, while pattern description components manage the accompanying information on pattern applications.

In the example of Figure 6 an SE component is shown by the *SE Community's Client Module* that accesses some third-party software, here a database management system storing the SE asset definitions.





among students can take place in the course component. Students can publish intermediate results to this component—at least if they meet all formal constraints—making them visible to students of the same course. These students can then, for example, annotate the proposed asset definitions.

Teachers, in addition to providing learning materials, take the role of a supervisor while students work on their tasks. This role is supported by an additional component accessible via *Supervisor Access*. This component does not maintain any additional content. Instead it allows a supervisor to take a look at both the materials presented to a course and the personalizations made by all participating students. This way a supervisor can advise students by monitoring their progress while given models can be inspected if required through the access to the course component.

#### 4. Summary and Outlook

We have shown that it is beneficial for teaching purposes to model design patterns dualistically. This helps students to understand patterns by rich medial representation of software design as well as a conceptual models pointing out their particularities. Furthermore schema personalization enables active learning as it allows learners to model the subject under study in the most appropriate way. Similar benefits arise from instance personalization where students are asked to complete partial content.

In future work it will be interesting to extend our conceptual model of design patterns to reach further into SE. It can then not only be used for teaching but will also be applicable to software creation proper. In the area of learning systems improving the reactions of the system to its users seems promising. The goal is to make learning systems react smartly to student's modeling decisions. A promising approach is to try to detect common misconceptions with patterns. This way learners receive more directed feedback and do not need to consult their supervisor or fellow students in order to understand the basics of patterns.

#### References

- [1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company, New York, NY, 1994.
- [2] Ian Sommerville. *Software Engineering*. Addison-Wesley, 2000.
- [3] Hans-Werner Sehring, Sebastian Bossung, and Joachim W. Schmidt. Active Learning By Personalization - Lessons Learnt from Research in Conceptual Content Management. In *Proceedings of the 1st International Conference on Web Information Systems and Technologies*, pages 496–503. INSTICC Press Miami, May 2005.
- [4] Sebastian Bossung, Hans-Werner Sehring, Patrick Hupe, and Joachim W. Schmidt. Open and Dynamic Schema Evolution in Content-intensive Web Applications. In *Proceedings of the Second International Conference on Web Information Systems and Technologies*, pages 109–116. INSTICC, INSTICC Press, 2006.
- [5] Yann-Gaël Guéhéneuc, Stefan Monnier, and Giuliano Antoniol. Evaluating the Use of Design Patterns during Program Comprehension – Experimental Setting. In *Proceedings of the 1st International Workshop in Design Pattern Theory and Practice*. IEEE Computer Society Press, 2005.
- [6] Iris Reinhardt-Berger. Conceptual Modeling of Structure and Behavior with UML – The Top Level Object-Oriented Framework (TLOOF) Approach. In *Proceedings of the International Conference on Conceptual Modeling - ER 2005*, volume 3716 of LNCS, pages 1–15, 2005.

- [7] Sebastian Bossung, Hans-Werner Sehring, Michael Skusa, and Joachim W. Schmidt. Conceptual Content Management for Software Engineering Processes. In *Advances in Databases and Information Systems: 9th East European Conference, ADBIS 2005*, volume 3631 of *Lecture Notes in Computer Science*, page 309. Springer-Verlag, 2005.
- [8] Hans-Werner Sehring and Joachim W. Schmidt. Beyond Databases: An Asset Language for Conceptual Content Management. In *Proc. 8th East European Conference on Advances in Databases and Information Systems*, volume 3255 of *LNCS*, pages 99–112. Springer-Verlag, 2004.
- [9] Object Management Group. *Meta Object Facility (MOF) Specification*, 1.4.1 edition, July 2005.
- [10] Joachim W. Schmidt and Hans-Werner Sehring. Conceptual Content Modeling and Management: The Rationale of an Asset Language. In *Proceedings of the International Andrei Ershov Memorial Conference, Perspectives of System Informatics, PSI '03*, volume 2890 of *LNCS*, pages 469–493. Springer-Verlag, 2003.
- [11] John Vlissides. *Pattern Hatching: Design Patterns Applied*. The Software Pattern Series. Addison Wesley Longman, 1998.
- [12] Bernd-Uwe Pagel and Mario Winter. Towards Pattern-Based Tools. In *Proceedings of the European Conference on Pattern Languages of Programming and Computing '96*, 1996.
- [13] Gerson Sunye, Alain Le Guennec, and Jean-Marc Jézéquel. Design Patterns Application in UML. In *ECOOP*, pages 44–62, 2000.
- [14] Gert Florijn, Marco Meijers, and Pieter van Winsen. Tool Support for Object-oriented Patterns. In M. Aksit and S. Matsuo, editors, *Proceedings of ECOOP '97 - Object-Oriented Programming: 11th European Conference*, volume 1241 of *Lecture Notes in Computer Science*, page 472. Springer, 1997.
- [15] Yann-Gaël Guéhéneuc, Houari A. Sahraoui, and Farouk Zaidi. Fingerprinting Design Patterns. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004)*, pages 172–181. IEEE Computer Society, 2004.
- [16] Daniel Lucrédio, Alexandre Alvaro, Eduardo Santana de Almeida, and Antonio Francisco do Prado. MVCASE Tool – Working with Design Patterns. In *Proceedings of the Third Latin American Conference on Pattern Languages of Programming (SugarLoafPLOP 2003)*, 2003.
- [17] Hervé Albin-Amiot, Pierre Cointe, Yann-Gaël Guéhéneuc, and Narendra Jussien. Instantiating and Detecting Design Patterns: Putting Bits and Pieces Together. In *ASE*, pages 166–173. IEEE Computer Society, 2001.
- [18] Tommi Mikkonen. Formalizing Design Patterns. In *ICSE '98: Proceedings of the 20th international conference on Software engineering*, pages 115–124, Washington, DC, USA, 1998. IEEE Computer Society.
- [19] W.J. Clancy. A Tutorial on Situated Learning. In *Proceedings of the International Conference on Computers and Education*, 1995.
- [20] Ulrik Schroeder. Meta-Learning Functionality in eLearning Systems. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Education, Science, and Medicine on the Internet*, 2002.
- [21] Heidrun Allert, Christoph Richter, and Wolfgang Nejd. Lifelong Learning and Second-order Learning Objects. *British Journal of Educational Technology*, 35(6):701–715, 2004.
- [22] Sissel Guttomsen Schär and Helmut Krueger. Learning Technologies with Multimedia. *IEEE Multimedia*, 7(3):40–51, 2000.
- [23] M. Dertnl and R. Motschnig-Pitrik. Conceptual Modeling of Reusable Learning Scenarios for Person-Centered e-Learning. In *Proceedings of International Workshop for Interactive Computer-Aided Learning (ICL'03)*. Kassel University Press, 2003.
- [24] Hans-Werner Sehring, Sebastian Bossung, and Joachim W. Schmidt. Content is Capricious: A Case for Dynamic System Generation. In Yannis Manolopoulos, Jaroslav Pokorný, and Timos Sellis, editors, *Proceedings of Advances on Databases and Information Systems: 10th East European Conference, ADBIS 2006*, volume 4152 of *LNCS*, pages 430–445. Springer-Verlag, 2006.
- [25] G. Meszaros and J. Doble. Metapatterns: A Pattern Language for Pattern Writing. In *Thrid Pattern Languages of Programming Conference*, Monticello, USA, 1996.

# Business Modeling and Enterprise Engineering

This page intentionally left blank

# A Technique for the Prediction of Deadline-Violations in Inter-Organizational Business Processes

Johann EDER,<sup>a</sup> Horst PICHLER<sup>b</sup> and Stefan VIELGUT<sup>b</sup>

<sup>a</sup> *University of Vienna, Department of Knowledge and Business Engineering, Austria*

<sup>b</sup> *University of Klagenfurt, Department of Informatics-Systems, Austria*

**Abstract.** Workflow time management provides predictive features to forecast eventually upcoming deadline violations and proactive strategies to speed up late processes. Existing time management approaches assume that communication with external processes or services is conducted synchronously. This is not the case with inter-organizational processes which very frequently communicate in an asynchronous manner. Therefore we examine diverse asynchronous communication patterns, show how to map them on an interval-based time model, and describe their application to inter-organizational workflow environments.

**Keywords.** Inter-organizational, business process, workflow, time management, deadline-avoidance, asynchronous communication.

## Introduction

Workflow management systems are used to improve business processes by automating tasks and getting the right information to the right place for a specific job function [1]. Nowadays, as business processes spread over the boundaries of companies, workflows are assembled from external processes and (web-)services [2]. They connect companies with their suppliers and providers for the achievement of inter-organizational business goals. More than ever services with a high quality must be provided, where expected process execution times and compliance to agreed upon deadlines rank among the most important quality measures [3]. However, unexpected delays can lead to time violations. This typically increases the execution time and cost of business processes because they require some type of exception handling, entail displeased business partners and customers, or even results in penalty payments.

To decrease the number of deadline violations should therefore be one of the major objectives, which can be achieved by the application of workflow time management. It deals with temporal aspects of time-constrained processes and aims at optimized, timely, and violation-free process execution. Predictive techniques are used to determine the remaining process duration and the expected end of a process. This is utilized to forecast upcoming deadline violations, and to trigger automatic evasive actions in order to avoid them, e.g. skip unnecessary optional tasks, increase resources, and so on (e.g. [4,5]). The calculation of the process remaining duration at workflow build-time is based on explicit

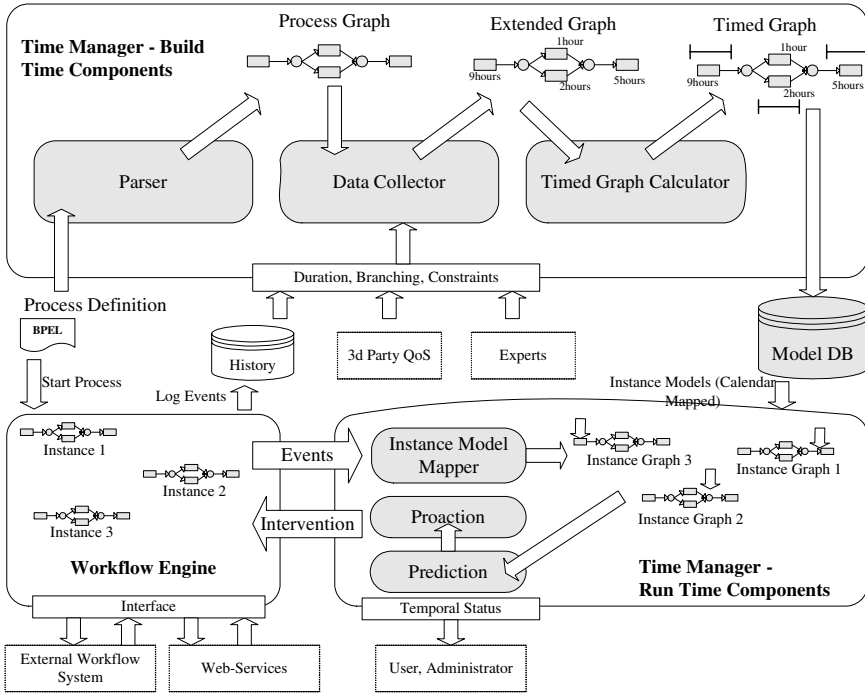


Figure 1. Time manager architecture (adapted from [7]).

knowledge about the process structure and the expected duration of each step. Unfortunately asynchronous (non-blocking) communication patterns are not supported by existing time management approaches. As inter-organizational business process communication is quite frequently asynchronous, it is a necessity to examine basic asynchronous process communication patterns, show how they affect the process execution and duration, and adapt existing time management techniques correspondingly.

In [6,7,8] we already provided partial solutions to certain time management problems. This paper combines and completes our prior findings as follow: in Section 1 we sketch components, architecture, and functionality of a time-managed workflow system. Section 2 describes a graph-based workflow model that is augmented with explicit timing information, followed by Section 3, which defines an interval-based method to calculate remaining times for each activity in the workflow. Sections 4 and 5 explain problems that arise when different processes communicate, list basic communication patterns, and show how to integrate them in the calculation algorithms. Finally we discuss related work in Section 7, followed by a brief outlook and conclusions in Section 8.

## 1. Time Management Architecture

Figure 1 visualizes how a time manager can be integrated into a workflow engine. Our architecture consists of the *workflow engine* and the time manager's *build* and *run time* components.

**Time Manager Build Time Components** The *parser* loads the process definition, parses it, and generates an according *workflow graph* that reflects the control flow of the process. The *data collector* augments the graph with additional temporal process information, like expected activity durations and deadlines. Information about the duration of external processes and services can be requested from third parties or experts. The resulting *extended graph* is fed into the *timed graph calculator*, which generates the *timed graph* [9]. The timed graph contains information about latest allowed end times for activities, that must not be exceeded during process execution – it is stored in the *model database* for run-time purposes.

**Workflow Engine** The *workflow engine* starts new process instances, controls their execution, and conducts communication with external services or processes [1]. During the execution of process instances certain events, like start or termination of processes and activities, are logged in the workflow history and signaled to the run time component of the time manager. In order to avoid possible future deadline violations the process engine reacts to special intervention signals from the time manager.

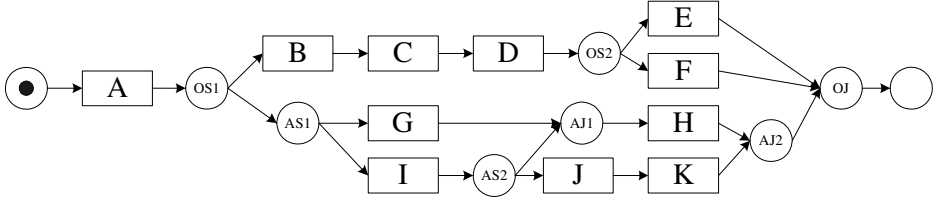
**Time Manager Run Time Components** When a process is started by the process engine, an according signal will be sent to the time manager. The *instance-model mapper* loads the corresponding timed graph from the model database and generates a *calendar-mapped* copy, called *timed instance graph*, for the process instance. Calendar-mapping means to map relative time values, stored in a timed graph, to the absolute (actual) system time. Each time an activity of the process instance starts or ends an according signal will be sent to the instance-model mapper.

- The *prediction component* periodically checks the temporal status of each process instance. This includes the expected process remaining duration, which may be accessed by users, service requestors or process administrators at any time. The remaining duration is utilized to forecast the expected end of the process, and if a deadline is likely to be violated.
- In case a possible deadline violation has been predicted the *proactive component* jumps into action, automatically initiates a pro-active avoiding action (e.g. rescheduling, exchanging upcoming activities with faster variants) in order to prevent a future deadline violation, and sends corresponding intervention instructions to the process engine.

## 2. A Workflow Model

### 2.1. Workflow Graph

A workflow basically describes a structured process consisting of activities and control flow dependencies between them (see also [1]). Figure 2 visualizes the graph-oriented representation of a process. Such a non-cyclic directed workflow graph  $G = (N, E)$  consists of a set of nodes  $N$  (activities or control-nodes) and a set of edges  $E$ . The type of a node  $n \in N$  can either be  $n.t = \text{activity}$  or one of the control-types  $n.t = \text{start} \mid \text{end} \mid \text{or-split} \mid \text{or-join} \mid \text{and-split} \mid \text{and-join}$ . Activities, represented by rectangles, correspond to individual steps in a process. A control flow dependency  $(n_1, n_2) \in E$ , displayed as edge, determines the execution sequence of two nodes  $n_1, n_2 \in N$ , such that  $n_1$



**Figure 2.** A sound workflow graph.

must be finished before  $n_2$  can be started. Control nodes, visualized as circles, represent workflow control structures. The label of a control node identifies its type. A circle with a dot depicts the start of a workflow and an empty circle depicts its end. A circle labeled with OS represents an or-split, which decides, based on a run time evaluated condition, which successor node will be executed next. In our model this node uses xor-semantics, as exactly one successor must be selected. The or-join, labeled with OJ, marks the end of an or-structure and allows the continuation as soon as one predecessor is finished. And-splits and and-joins, labeled with AS and AJ, are used to define parallel execution of several branches. All branches after the and-split will be executed in parallel and the and-join synchronizes all these branches, such that continuation is allowed if and only if all its predecessor nodes are finished.

## 2.2. Structural Constraints

We demand that the workflow graph has exactly one start and one end-node. Additionally only split-nodes may have multiple successors and only join-nodes may have multiple predecessors. The structure of the graph may be *non-blocked* [1] and must be *sound*. In non-blocked structures edges may connect activities and/or control-nodes in an arbitrary (non-cyclic) order. This can result in process structures that produce deadlocks (e.g. or-split closed by an and-join) or unwanted multiple instances (e.g. and-split closed by an or-join) during process execution. The concept of *soundness* restricts combination-possibilities of control-flow elements to certain patterns, which naturally constrains the degree of modelling freedom, but also eliminates the possibility of above-mentioned runtime problems. For further discussions on the soundness of workflows please refer for instance to [10]. Additionally we stated that the workflow graph must be non-cyclic, but the possibility to model cyclic structures is a necessity for the automatization of business processes. Therefore we propose to apply one of the following alternatives which may be used for any blocked cyclic structure<sup>1</sup>: a) hide the whole cycle in a single activity; b) roll the cycle out to a sequence by means of the expected (average) number of iterations; or c) unfold the cycle to a conditional structure by inserting an or-split after each iteration-block, where one edge connects the or-split with the next iteration-block, and the other edge connects it with the control node that closes the cyclic structure [6].

<sup>1</sup>Blocked loops are for instance *while*, *repeat*, or *loop*-type cycles. Arbitrary cycles [10], which allow interleaved iterations, are not considered here.



### 2.3. Extended Workflow Graph

In order to apply time management techniques it is necessary to augment the basic workflow graph with explicit temporal information.

- It is necessary to provide an expected duration for each activity. Following [11,9, 12,13] we apply an interval-based representation, which describes a duration by means of a  $[\min, \max]$ -interval. From now on the duration of any node  $n \in \mathcal{N}$  is denoted by an interval  $n.d = [a, b]$ , which implies that the duration of  $n$  will presumably not fall below  $a$  and presumably not exceed  $b$ . These values are given in a predefined basic time-unit, like minutes or hours.
- Second, the process is constrained by a maximum duration  $\delta$ , also called *relative deadline*, which must not be exceeded by any process instance. Again this value must be specified in the above-mentioned predefined basic time-unit.

The information about durations may stem from empirical knowledge (extracted from the workflow history, where past process executions are logged), from experts, or even from third parties in case an activity communicates with an external process or service. The deadline stems from laws, organizational rules, or contracts placed with customers (which may also include penalty payments on deadline-exceedance).

## 3. Timed Workflow Graph

For predictive or proactive time management [4,14] we have to calculate additional time information for each node at build time, and store it in the so-called *timed workflow graph* for further usage during run time. The temporal information we need, namely remaining times, must be calculated for each node, by utilizing the precedence constraints of the structure and the explicit temporal information provided by the extended graph. The remaining-time interval  $n.re = [a, b]$  of a node  $n \in \mathcal{N}$  denotes the expected execution time between the (start of the) node and the end of the process. For the calculation of the overall process remaining duration we have to add up remaining times – therefore we define the addition of two time intervals as follows:

**Definition 1** *Interval Addition:*  $[a_1, b_1] + [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)]$

To calculate the remaining time of each node in the graph, we must apply the calculation operations specified below on each node  $n$ , dependent on its type  $n.t$ , in a backward topological order, starting with the last node in the process.

#### 3.1. End-node

The calculation starts with the initialization of the end-node  $n$  with  $n.re = n.d$ , which simply means that the remaining time, measured from the the start of the last node, is equal to the duration of this node.

#### 3.2. Activity, or-join, and-join, start-node

The remaining-time interval of a node  $n$ , which has exactly one successor  $s$ , is calculated as  $n.re = s.re + n.d$ , where  $(n, s) \in E$ .

### 3.3. Or-split

The remaining time of an or-split is based on the interval-disjunction operation.

**Definition 2** *Interval Disjunction:*  $[a_1, b_1] \vee [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)]$ .

*Proposition:* As the disjunction is commutative and associative it can be extended to  $j$  intervals  $d_1 \vee d_2 \vee \dots \vee d_j$ , denoted as  $\bigvee_{i=1}^j d_i$ .

The remaining time of an or-split  $n$  can now be calculated as  $n.re = (\bigvee_{i=1}^j s_i.re) + n.d$ , where  $(n, s_1), \dots, (n, s_j) \in E$ . In the first step the disjunction-operation merges all remaining-time intervals  $[a_i, b_i]$  of all successors  $s_i$ , such that the resulting interval is determined by the smallest  $a_i$  and the greatest  $b_i$ . This is necessary as only one out of many paths will be followed after an or-split, and which one this will be is not known at build time – therefore one can only consider the best and the worst case. Then the duration-interval  $n.d$  of the or-split must be added before proceeding with the calculation of predecessor nodes.

### 3.4. And-split

The remaining time of an and-split is based on the interval-conjunction operation.

**Definition 3** *Interval Conjunction:*  $[a_1, b_1] \wedge [a_2, b_2] = [\max(a_1, a_2), \min(b_1, b_2)]$ .

*Proposition:* As the conjunction is commutative and associative it can be extended to  $j$  intervals  $d_1 \wedge d_2 \wedge \dots \wedge d_j$ , denoted as  $\bigwedge_{i=1}^j d_i$ .

The remaining time of an and-split  $n$  can now be calculated as  $n.re = (\bigwedge_{i=1}^j s_i.re) + n.d$ , where  $(n, s_1), \dots, (n, s_j) \in E$ . In the first step the conjunction-operation merges all remaining-time intervals  $[a_i, b_i]$  of all successors  $s_i$ , such that the resulting interval is determined by the greatest  $\min_i$  and the greatest  $\max_i$ . This is necessary as every path succeeding an and-split will be executed in parallel – therefore one must consider the worst case. Again the duration-interval  $n.d$  of the and-split must be added before proceeding with the calculation of predecessor nodes.

## 4. Communication with External Processes and Services

In inter-organizational scenarios one will be confronted with the situation that external services, applications, or sub-processes are called from the local main process. The processes will communicate in a blocking (synchronous) or non-blocking (asynchronous) manner, which raises problems for the calculation of remaining times.

### 4.1. Synchronous and Asynchronous Communication

In a *synchronous* or *blocking* model the requester waits for the response of the provider before continuing execution. The advantage of this model is its simplicity, as the process state does not change until the response has been received. The obvious disadvantage is that blocking the execution of the main process, especially when long running external processes or services are involved, increases its execution duration tremendously.

In an *asynchronous* or *non-blocking* model the requester sends a request to the provider and continues execution without delay. At a later point in time it receives a response (callback) from the provider, which of course implies that the main process contains an activity that waits to receive this response. Asynchronous communication loosely couples sender and receiver. This accelerates process execution and compensates communication problems (e.g. network problems).

#### 4.2. Implications on Time Management

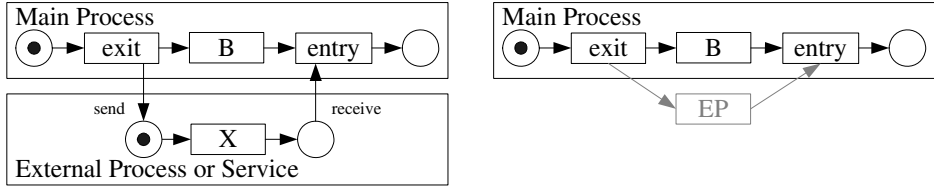
So far existing time management approaches are based on one assumption: activities, may they be atomic or composite, are interpreted as basic execution units which must be finished in order to proceed with the process instance. This still applies for synchronous, but not for asynchronous communication. In order to explain the integration of these models into our time management calculation algorithms, we have to clarify what we focus on. We are interested in the effects different types of communication patterns have on the duration of the main process and the delay they might produce between different activities of the very same. Please note that we calculate our time models for the main process only, therefore our primary interest in external services or processes is their *response time*. For time management it is not important to know how communication works in process automation systems. We assume that the system relies on a communication infrastructure, which forwards request-messages from the main process and receives response-messages from external processes. This may be implemented by means of message queues [2]. Furthermore we do not differ between communication, execution or waiting time of external services; we use an overall response time which suffices for our needs. As messages are queued they may be retrieved before they are needed in the process. Therefore it is important to notice that the response time is measured as the time span between the outgoing message leaving the out-queue and the correlating ingoing message entering the in-queue.<sup>2</sup> Response times may be estimated or gathered from empirical knowledge, for instance extracted from the message-log which holds information about prior communication-sequences or from a third party (see [3,15]).

#### 4.3. Solution

Consider the process on the left-hand side of Figure 3, which shows a standard asynchronous communication scenario. Activity *exit* sends a message which activates an external process, while the main process continues with its execution. Assuming that the response time of the external process is longer than the duration of the activity *B*, the receiving activity *entry* will have to wait for the results of the external process in order to proceed. As a matter of fact this scenario is equal to a regular parallel structure. Since the external process may reside anywhere, and therefore its structure may be unknown, it must be treated as a black box – therefore it is necessary that the process designer introduces a corresponding *virtual* activity *EP* into the workflow graph. Then he has to specify additional edges, which connect the exit-node, via the virtual activity, with the entry-node (right-hand side of Figure 3). Additionally required is the expected execution duration interval for the virtual activity, determined by the response time of the external

---

<sup>2</sup>For details on the implementation of callbacks and the correlation of messages with their appropriate process instances, for instance with correlation-ids, we refer to [2].



**Figure 3.** Communication with an external process.



**Figure 4.** Mapping of synchronous patterns.

process. The duration of the exit-node is determined by the time that it takes to pass the request-message to into the out-queue, and the duration of the entry-node is determined by the time that it takes to fetch a message from the in-queue – both durations will presumably be neglectable compared to durations of regular activities. Time management calculations are to be applied by treating the exit-node like an and-split and the entry-node like an and-join. Correspondingly the same structural restrictions and soundness criterions as for regular parallel structures must be applied. It is for instance not allowed to place an exit-node on a conditional branch, and the adhering entry-node on a branch that will always be executed – this may result in a deadlock as it is possible that the entry-node waits for a response that never comes, as no prior request has been sent.

## 5. Examination of Communication Patterns

Although the above presented solution works for this special scenario, it is still necessary to identify further communication-scenarios and examine how they affect the calculation-operations. Recent publications on web service communication and web service composition (e.g. [2,16]) already identified several basic synchronous and asynchronous communication patterns, which we utilized for our purposes. Note that we explain these patterns in the context of a local main process that sends and receives messages to and from an external process respectively. In the following the identifiers of synchronous patterns are pre-fixed with *SCP* and those of asynchronous patterns with *ACP*. Figures 4, 5, 6 and 7 visualize necessary mappings for an application in a workflow graph.

### 5.1. SCP1 - Request/Reply

Request/Reply is the basic model for synchronous communication. As visualized in Figure 4 the main process sends a request to the external process and blocks until a response is returned. Therefore it is not allowed to place further nodes between the exit and

the entry-node. Otherwise it can be treated like the standard asynchronous scenario explained above. This pattern may also be represented as a *complex* activity [14]. A complex activity hides nested process structures, in this case the exit and the entry-node. Its duration is defined by the response time of the external process.

### 5.2. SCP2 - Solicit Response

A solicit response is an inverted SCP1, where the external process acts as requestor and the main process as provider (see Figure 4). The main process must wait for a message from an external process at the entry point in order to process the request. As this pattern assumes that no prior request has been sent from the main process, the duration of the entry-node must be specified by waiting-time interval, which is determined by the expected time span the entry-node has to wait for the incoming message. The succeeding activity *A* stands as placeholder for an arbitrary number of activities or control flow structures. The exit-node sends the response to the requesting external process. For time management calculations it is not necessary to know that the entry-node and the exit-node adhere to the same communication sequence, therefore no connecting construct is needed. This pattern is unlikely to occur in a driving main process, which usually treats other processes as mere service providers.

### 5.3. SCP3 - Synchronous Polling

This pattern is an extension of SCP1. Again the main process sends a request and blocks. Subsequently it checks in defined intervals until the response arrives, then it stops further polling-attempts and proceeds with execution. For time management calculations it is not important to know the number of polling attempts. The only information needed is again the response time, which is the time span between the original request and the last successful polling attempt. Therefore it can be treated like SCP1.

### 5.4. ACP1 - One-way (Message Passing)

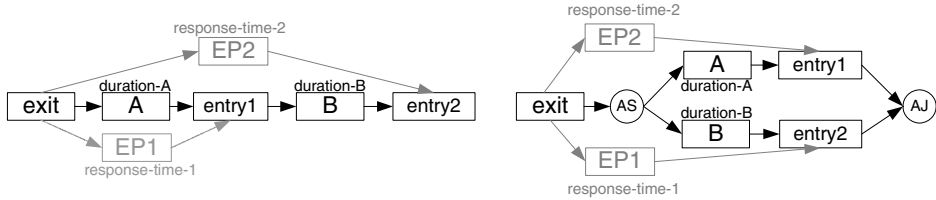
The main process sends a message to the external process and since it does not expect a reply it immediately continues with execution. This pattern requires no synchronization.

### 5.5. ACP2 - Notification

This pattern is an inverted ACP1. The external process sends a message to the main process, without expecting a response. Alike SCP2 this pattern assumes that no prior request has been sent, and can therefore be treated analogously. Again we assume that this pattern is unlikely to occur in a driving main process, which usually treats other processes as mere service providers.

### 5.6. ACP3 - Request/Response

This pattern is the combination of ACP1 and ACP2. Scenario and the corresponding solution have already been presented in Section 4.3.



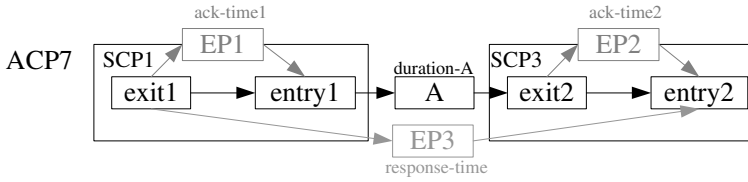
**Figure 5.** Mapping of pattern ACP4.

### 5.7. ACP4 - Request/Multiple Response

This pattern is an extended ACP3, as visualized in Figure 5. The main process sends a message to the external process, which returns several notifications at different points in time. As the process does not block execution after sending a message, there may be an arbitrary number of activities, represented by *A*, after the exit-node. But now the main process expects multiple responses from the external process, which are received in multiple entry-nodes (see left-hand side of Figure 5). Therefore a virtual activity must be inserted between the exit-node and each entry-node. Another possibility is visualized on the right-hand side of the figure. Here the messages are received in parallel branches of the main process; although the structure is different, the concept of inserting a virtual activity between each combination does not change (the same applies if the and-structure is exchanged by an or-structure). The determination of a response time for each combination poses no problem if an entry-node expects a specific type of message, which can only be received by this entry-node. If some or even all entry-nodes are implemented to receive the same type of message an average response time may be used for each virtual activity.

### 5.8. ACP5 - Publish/Subscribe

In a publish-subscribe system the sender augments each message with a topic. The messaging-system forwards the message to all subscribers that have asked to receive messages on that topic. Senders are not burdened with creation and dispatching of message-duplicates to each subscriber. This is done by the messaging infrastructure. Publish-subscribe is a very loosely coupled architecture, in which senders do not need to know who their subscribers are. Assuming that the main process is the subscriber, this pattern can be mapped using a combination of other patterns. With SCP1 or ACP1 we model the subscription, depending on whether the main process has to wait for acknowledgement or not. ACP2 can be used for each point in the main process, where a publication message is expected. The waiting time for ACP2 is the average time span between two publications. Of course it makes sense to differentiate between waiting times for specific topics, in case it is known which message to expect at an entry-node. In our opinion this pattern is unlikely to appear in the course of a business process. It may probably be used to start a process when such a message is received, which has no influence on time management calculations.



**Figure 6.** Mapping of pattern ACP7.

### 5.9. ACP6 - Broadcast

Broadcast can be interpreted as an inverted ACP5. A message is sent to the messaging infrastructure, which delivers a message to a list of subscribed receivers. Each of the receivers decides how to further process this message. The sender does not expect any acknowledging responses. Similar to ACP5 broadcast may significantly increase network traffic, therefore it should be used carefully and for very specific purposes only. The broadcast is mapped by applying a single exit-node, like in ACP1, as the selection of receivers and sending multiple messages is conducted by the messaging system.

### 5.10. ACP7 - Request/Response with polling

This pattern is used if callbacks from external processes are not possible or allowed. The main process calls an external process using a blocking SCP1 Request/Reply, where the reply is an acknowledged-message from the receiver. Afterwards the main process continues. At a later point in time, when the main process needs the results, it calls the external process again; this time it uses SCP3 synchronous polling until it receives the response. Subsequently the main process may proceed. The first request with acknowledgement can be modelled with SCP1 request/reply. When the main process needs the results it starts polling the external process, which is modelled using a SCP3 request/reply with polling – the duration of the corresponding virtual activity comprises all polling attempts. The response time, representing the actual execution time needed by the external service to process the primary request, must be introduced as duration of the virtual activity between the first exit and the second entry-node. At the second entry-node the final response is received by the main-process. Alternatively SCP1 and SCP3 may be represented as complex activities, as they are blocking anyway.

### 5.11. ACP8 - Request/Response with posting

This pattern is applied for business-critical or confidential messages. As visualized in Figure 7 it consists of a SCP1 request/reply followed by SCP2 solicit response. The main process sends a message to the external process, blocks and waits for the immediate acknowledging response. Later, the external process sends an answer which must again be immediately acknowledged by the main-process. Again the first request with acknowledgement is modelled using SC1 request/reply. After the execution of A, the main process has to wait for the response from the external process before it can return an acknowledged-message. Therefore SCP2', a solicit response variant, is used. The (complex) activity B may among other things for instance generate the acknowledged-

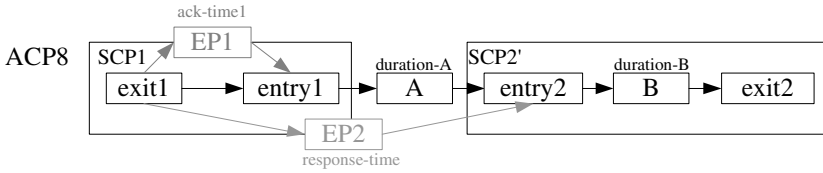


Figure 7. Mapping of pattern ACP8.

message. The difference between SCP2' and the original SCP2 is that the entry node is not augmented with an average waiting time. The response message will be received when the external process finished its execution. Therefore it is sufficient to introduce a virtual activity between the primary exit- and the second entry-node augmented with the response time, which represents the duration of the external process. Finally the main process will send an acknowledged-message at the second exit-node.

## 6. Application at Run Time

The timed graph, already stored in the model database, is utilized during run time to assess the current temporal state of a process instances. On start of a new process instance the corresponding timed graph must be loaded from the model database, and, as the execution proceeds, this instance must be synchronized with the timed graph model. The process deadline is specified as relative value; therefor it must be adjusted (calendar-mapped) to the current system time *now*:  $\delta' = \delta + \text{now}$ . This version of the timed graph is called *instance graph*, as it adheres to one specific process instance.

### 6.1. Prediction of Process Remaining Duration

To determine the expected process remaining duration one has to adjust the remaining-time interval  $n.re = [a, b]$  of the currently active node  $n$ , since the already elapsed execution duration  $\sigma$  of  $n$  must be taken into account:  $n.re' = [a - \sigma, b - \sigma]$ . Given that we know the current time *now*, the end of the process can therefor be expected between  $\Omega = [a' + \text{now}, b' + \text{now}]$ . Note that it is quite simple to determine the elapsed execution duration of the current node, since a workflow system usually logs all kinds of events, including the *start-time* of this node:  $\sigma = \text{now} - \text{start}$ . The intervals for the expected process remaining duration as well as the expected end of the process can be provided to participants, administrators or customers.

### 6.2. On Parallel Execution

The semantics of a parallel structure (and-split) demands that each path started by it behaves like an independent sub-process, until merged into the main process by an and-join. This implies that the state of execution of such a parallel sub-process is also independent from all its siblings – accordingly several execution tokens exist in a single process instance, one for each sub-process, and each of them progresses independently from the others. According to the semantics of remaining-times, each activity that resides on



a parallel path is treated as if it were executed in isolation from its parallel siblings, only influenced by precedence constraints. This poses a problem for the determination of the process remaining duration, where the point of view is the whole process and not a single activity, as all currently active siblings must be taken into considerations. In case  $j$  nodes  $n_i, 1 \leq i \leq j$  are executed in parallel the process remaining duration must be calculated in two steps: first we determine the worst case for all currently active parallel nodes  $worst = (\bigwedge_{i=1}^j n_i.re')$ . Then, given that  $worst = [a, b]$ , the process remaining duration can be determined by  $\Omega = [a + now, b + now]$ .

### 6.3. Prediction of Deadline Violations

The prediction of eventually upcoming deadline violations, based on the adjusted remaining-time interval  $n.re' = [a', b']$  and the calendar-mapped deadline  $\delta'$ , is best explained by means of a simple example: assume that at  $now = 8:10 a.m.$  the node  $n$  is active, that  $n.re' = [15, 35]$ , and the basic time-unit is *minutes*. The end of the process can therefor be expected between  $\Omega = [a' + now, b' + now] = [8:25, 8:45]$ . Further assume that the calendar-mapped process deadline is  $\delta' = 8:20$ . According to the expected end, one can state that this deadline will not hold, even if the fastest path will be selected at each or-split. Now assume that the deadline is  $\delta' = 8:50$ ; in this case the process instance still has enough buffer time – this means that it bears a delay of at least 5 minutes without endangering the deadline.

To assess the current temporal state for further proactive usage, we adjust the *traffic light model* proposed by [14]. The traffic light is basically an urgency-indicator; it switches the (temporal) state of a process instance according to the likelihood of a future deadline violation. For our interval-timed model it is applied, by utilizing  $\delta'$  and  $\Omega = [a' + now, b' + now]$ , as follows:

- Green** The process instance is in state green if  $\delta' \leq a' + now$ , which means that the instance is fast enough and will most certainly meet the deadline.
- Orange** The process instance is in state orange if  $\delta' \leq b' + now$ , which means that one should have an eye on this instance, as the deadline may be violated if paths with long durations are selected in the future. Proactive strategies may be applied if  $\delta'$  converges  $b' + now$ .
- Red** The process instance is in state red if  $\delta' > b' + now$ , which means that the process deadline will most likely be violated. Proactive strategies must be applied to avoid a deadline violation.

### 6.4. Proactive Strategies

In case the proactive component catches an exception from the prediction component, the process instance is already late. This means that the rest of the instance must be sped-up in order to reach the given deadline. In e.g. [4,9,12,5] diverse automated proactive escalation alternatives are discussed. This comprises for instance to skip unnecessary optional tasks, to parallelize activities that are normally executed sequentially, to add further resources (man-power), to increase the priority of late process instances, and so on. The problem is that many of these strategies are not applicable in inter-organizational scenarios, as the workflow system does not have the means to influence the execution of external services, called by activities of its process instances. Therefor additional strate-

gies must be found. We just started to investigate a technique that aim at exchanging services, to be called by future activities, with faster alternatives (with a lower expected duration) during process execution. Of course it is possible that these alternative services have some drawbacks (which is the reason that they were not chosen in the first place), e.g. they might be more expensive. The first input parameter for such an algorithm is the amount of time that must be saved, which can be easily provided with our approach. Additionally the algorithm needs to know which services are exchangeable, along with a set of alternative services for each of them (for the time being we plan to realize a static approach). The next step is the generation of an alternative process execution plan. This is not as straightforward as it seems at first glance. E.g. exchanging a service with a shorter alternative might not affect the execution duration of the process at all, if it for example resides on a path where still slack time is available. Additionally it might be necessary to exchange more than one service. However, in every case a partial recalculation of the timed instance graph will be necessary. Finally the process engine must be informed about the changes it has to apply on the (still running) process instance.

## **7. Related Work**

Asynchronous process communication patterns, applied in this paper, are described in [2,16]. The basic concepts for the calculation of a timed graph are rooted in project planning methods (PPM), like the Critical Path Method (CPM) or the Program Evaluation and Review Technique (PERT), extended for basic workflow structures by various authors (e.g. [11,12,9,14,17]). The usage of interval-timed models is very popular in time management literature and quite frequently applied in PPM-based methods (e.g. [12,9]), as well as in methods that originate from temporal constraint networks (e.g. [11,17]). However, none of them deals with asynchronous process communication. The calculation of remaining-times, as presented in this paper, is a variation of the interval-based backward-calculation technique described in [9], adjusted for the calculation of remaining times (introduced in [6]), and extended for the usage of duration-intervals, as the original technique was designed for average durations. Please note that our solutions for asynchronous communication patterns do not depend on the interval-representation, they can basically be applied on any time management technique that supports parallel execution.

## **8. Conclusions and Future Work**

The prediction and proactive avoidance of deadline violations decreases costs of processes and increases their quality of service. In this paper we examined basic asynchronous process communication patterns frequently used in inter-organizational processes, showed how they affect an interval-timed workflow graph, and provided according extensions for our workflow time management algorithms. Currently we search for further communication patterns, aim for an application of time management in web service environments, and develop pro-active strategies that dynamically exchange services, to be called during process execution, in order to speed up the process.

## Acknowledgements

Part of this work has been supported by EU Commission within the FET-STREP Project WS-Diamond.

## References

- [1] *Workflow Process Definition Interface*. A Workflow Management Coalition (WfMC) specification, <http://www.wfmc.org>, document number WfMC-TC-1025, 2002.
- [2] G. Alonso, F. Casati, H. Kuno, V. Machiraju. *Web Services: Concepts, Architectures and Applications*, Springer Verlag, 2005.
- [3] M. Gillmann, G. Weikum, W. Wonner. Workflow Management with Service Quality Guarantees. *Proceedings of ACM SIGMOD International Conference on Management of Data*, ACM Press, 2002.
- [4] E. Panagos, M. Rabinovich. Predictive Workflow Management. *Proceedings of the Third International Workshop on Next Generation Information Technologies and Systems*, Neve Ilan, Israel, 1997.
- [5] W.M.P. van der Aalst, M. Rosemann, M. Dumas. Deadline-based Escalation in Process-Aware Information Systems. *BPM Center Report BPM-05-05*, BPMcenter.org, 2005.
- [6] J. Eder, H. Pichler. Duration Histograms for Workflow Systems. *Proceedings of the IFIP TC8/WG8.1 Working Conference on Engineering Information Systems in the Internet Context*, Kluwer Publishers, 2002.
- [7] J. Eder, H. Pichler, S. Vielgut. An Architecture for Proactive Timed Web Service Compositions. *Business Process Management Workshops (Proceedings)*, LNCS 4103, Springer Verlag, 2006.
- [8] J. Eder, H. Pichler, S. Vielgut. Avoidance of Deadline-violations for Inter-organizational Business Processes. *Proceedings of the Seventh International Baltic Conference Databases and Information Systems*, IEEE Computer Society Press, 2006.
- [9] J. Eder, E. Panagos, M. Rabinovich. Time Constraints in Workflow Systems. *Proceedings of the 11th International Conference on Advanced Information Systems Engineering*, LNCS 1626, Springer Verlag, 1999.
- [10] B. Kiepuszewski, A. ter Hofstede, C. Bussler. On Structured Workflow Modeling. *Advanced Information Systems Engineering: 12th International Conference (CAiSE 2000)*, *Proceedings*, LNCS 1789, Springer Verlag, 2000.
- [11] I. J. Haimowitz, J. Farley, G. S. Fields, J. Stillman, B. J. Vivier. Temporal Reasoning for Automated Workflow in Health Care Enterprises. *Electronic Commerce, Current Research Issues and Applications (Workshop at NIST)*, LNCS 1028, Springer Verlag, 1996.
- [12] O. Marjanovic, M. Orlowska. On Modeling and Verification of Temporal Constraints in Production Workflows. *Knowledge and Information Systems*, 1(2), 1999.
- [13] G. Baggio, J. Wainer, C. A. Ellis. Applying Scheduling Techniques to Minimize the Number of Late Jobs in Workflow Systems. *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC)*, ACM Press, 2004.
- [14] J. Eder and E. Panagos. Managing Time in Workflow Systems. *Workflow Handbook 2001*, Future Strategies Inc. Publ. in association with Workflow Management Coalition (WfMC), 2001.
- [15] J. Cardoso, A. Sheth, J. Miller. Workflow Quality of Service. *Proceedings of the International Conference on Integration and Modeling Technology and International Enterprise Modeling Conference (IEIMT/EMC)*, Kluwer Publishers, 2002.
- [16] E. Newcomer. *Understanding Web Services*, Addison-Wesley, 2002.
- [17] C. Bettini, X. S. Wang, S. Jajodia. Free Schedules for Free Agents in Workflow Systems. *Proceedings of the Seventh International Workshop on Temporal Representation and Reasoning (TIME 2000)*, IEEE Computer Society Press, 2000.

# Interactions at the Enterprise Knowledge Management Layer

Saulius GUDAS <sup>a,b</sup> and Rasa BRUNDZAITE <sup>a</sup>

<sup>a</sup> *Vilnius University, Kaunas Faculty of Humanities,*

*Muitines 8, LT-44280 Kaunas, Lithuania*

<sup>b</sup> *Kaunas University of Technology, Information Systems Department,*

*Studentu 50, LT-51368 Kaunas, Lithuania*

**Abstract.** The paper deals with modeling of the enterprise knowledge management. The enterprise knowledge base together with explicitly modeled knowledge management activity is concerned as the major component of the knowledge-based enterprise. The Knowledge-Based Enterprise Model is developed for the analysis and design of knowledge management activity domain as well as for modeling its relationships with business domain and information technology domain. Knowledge management layer is identified within the model and further is decomposed and represented as formalized interactions – knowledge management controls.

**Keywords.** Enterprise knowledge base, knowledge-based enterprise, knowledge management layer, knowledge management controls

## Introduction

Nowadays organizational knowledge itself, as well as knowledge manipulation activities and tools evoke a big interest both among business people and scientists. Knowledge manipulation activities, also known as Knowledge Life Cycle process, exist in almost every organization, “but the difference is the degree of its formalization, automation and optimization” [1]. Similarly to the business processes management techniques, the knowledge management (KM) methods and tools are required for the formalization, optimization, automation and management of the knowledge manipulation activities in the enterprise. Business process modeling (BPM) is at the core of the advanced business process management and information systems (IS) development methods. It seems that the modeling technique could be helpful for the optimization and automation of the knowledge manipulating and management activities in the organizations, i.e. for the development of the knowledge management system.

Contemporary business modeling methods deal mostly with the modeling of the knowledge manipulation activities and its relationships with the business process activities rather than modeling of knowledge management process and its connection with other business management activities. As the changes in the knowledge management processes evokes changes in the knowledge manipulation processes [1] and affects the business process as well as the information systems development process, the management aspects of the business and knowledge processes and their

relationships should not be neglected when implementing knowledge management systems in organizations.

The conception of the knowledge-based enterprise, presented in the article, embodies holistic view to the knowledge management system development. Presented approach comprises synergetic application of various modeling methods for the solution of the range of business problems: for optimization and computerization of the business process and knowledge manipulation processes as well as for computerization of their management processes; for business and IT alignment decisions making. The main component of presented approach is the enterprise knowledge base, which is the mandatory element of the knowledge-based enterprise. It is expected, that integration of an enterprise knowledge base in to the overall business management activities will improve the business management (a strategic, tactical and operational management), making it enterprise management system more adaptable and flexible.

In the first section conception of the knowledge based enterprise is explained and the aspects (domains) of enterprise knowledge are discussed.

The conceptual model of the knowledge-based enterprise (KBE) is presented in the second section. The KBE model is the background for the development of the enterprise knowledge base and knowledge management system. The KBE model is derived from the modified Porter's Value Chain (VCM) model [2] by redefining business process concept of VCM. The Knowledge management layer of the KBE model further is decomposed into hierarchical multilevel structure. The enterprise knowledge management components are formally represented as knowledge management controls.

## **1. The Conception of the Enterprise Knowledge Base**

According M. H. Zack, the concept of the knowledge-based enterprise has to be defined independently from the products/services produced by enterprise activities, but rather on how it interprets and manages its knowledge [3]. The knowledge-based enterprise here is defined as the more mature enterprise in the sense of knowledge management, which uses enterprise knowledge base integrated with KM activity at the all levels of business management hierarchy as the obligatory enterprise management and development component.

The research presented in [4] resulted in identification of three integrated aspects of the enterprise knowledge: business (V), information technologies (T) and knowledge (K). Consequently, enterprise knowledge base should comprise integrated knowledge about three enterprise domains into appropriate level of detail: business (V), information technology (T) and knowledge (K) as well as various relationships of these domains. S.Gudas, R.Brundzaite [5] have researched the levels of analysis of these domains and various relationships among these domains. The interactions of three domains in five levels of detail each were abstracted in the Enterprise Knowledge Space Model (VxTxK).

The concept of the enterprise knowledge base comes from the notion of the organizational or corporate memory found in the KM literature and from the notion of the enterprise repository, used in the knowledge-based IS engineering field.

The concept and the structure of the organizational memory in the KM literature are investigated. The organizational memory is concerned with the organizational learning processes and is considered as mandatory construct of any learning

organization. Organizational memory comprises all the possible forms of organizational knowledge: tacit, explicit, computerized, not computerized etc. Contemporary highly networked organizations use information technology extensively and needs for organizational memory accessible in the virtual environment in the computerized and well-structured form. This structured and computerized part of the organizational memory is termed here as enterprise knowledge base. S.Gudas, T.Skerys, A.Lopata [6] deal with knowledge-based enterprise from IT point of view and claims that the knowledge-based enterprise from this point of view has to be connected to the shared Enterprise Repository – the Enterprise Knowledge Base. Enterprise Repository supports all types of Enterprise activities - business management and IS engineering activities and stores validated enterprise knowledge in the form of enterprise models.

In order to assure integration of the Enterprise knowledge base into overall enterprise management and development framework, it is necessary to define the interactions of the enterprise knowledge base and knowledge management activities explicitly too. Research, presented in this article, is directed toward extension of the knowledge-based IS engineering paradigm, described by S.Gudas, T.Skerys, A.Lopata [6], by adding knowledge management dimension.

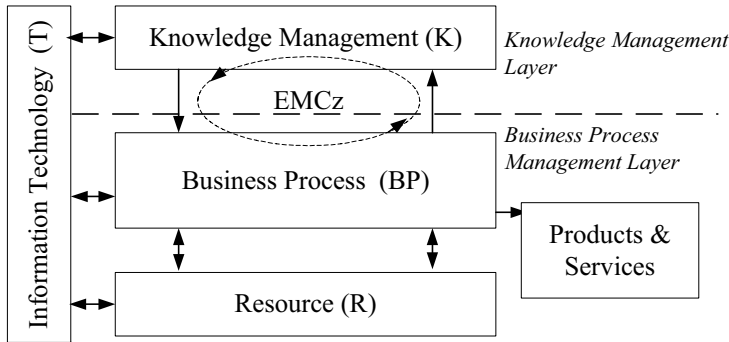
In order to develop practical method for the modeling of three interrelated enterprise domains (business, IT and knowledge), enterprise modeling method has to be selected. Contemporary enterprise modeling methods F3 [7], EKM [8]), etc. do not suit directly for the modeling of the three interrelated domains and their interactions. Knowledge-oriented enterprise modeling methods, such as (B-KIDE [9], Fraunhofer bpoKM (GPO-OM)) [10], and others, described in [11]) are directed toward modeling the knowledge manipulation activities (Knowledge Life Cycle) rather than knowledge management activities. Our point of view is that knowledge management activity is the central component of the knowledge-based enterprise model.

In the next chapter Knowledge-based enterprise model (KBEM) is presented. KBE model allows modeling of all three interrelated enterprise domains (business domain, IT domain and knowledge domain) as well as interactions among them.

## 2. Structural Backgrounds for Knowledge-Based Enterprise Modeling

The M. Porter's Value Chain Model (VCM) is used here as a basis for the enterprise knowledge modeling. The Porter's Value Chain Model is applied for business systems analysis, based on the separation between primary and support activities. Gudas, Lopata, Skerys [12] focused on the informational interrelationship between primary and support activities of the *Business Process* and identified different nature of these two kinds of activities: primary processes are non-informational and are named *Processes*; support activities have informational nature and referred to as *Process Management Functions*. The similar insights are represented in the Organizational Control Systems Modeling (OCSM) framework developed by Kampfner [13].

There is another type of business activity – knowledge management activity, which is also identified within modified VCM. For the completeness of the model the *Resources (R)* component was introduced. Whereas the modified Value Chain Model is focused on the enterprise knowledge management activities and components and it is constructed according knowledge-based enterprise definition, it is named Knowledge-Based Enterprise (KBE) model (Figure 1). The component *Business process (BP)* in Figure 1 consists of *Management functions (F)* and *Processes (P)*.



**Figure 1.** The Knowledge-Based Enterprise model

The KBE model refines two different levels of enterprise management hierarchy: *Knowledge management layer* and *Business process management layer*.

The interactions of the different levels are defined as two control loops (informational feedback) EMC (Elementary management cycle): EMCz and EMCp. EMC initially was introduced in [14] by Gudas and later was formally described in [12]. The semantics of the management transactions EMCp and EMCz are as follows:

- *Process management cycle (EMCp)* – this EMC implements a definite set of *Process management functions (F)* in Figure 3. EMCp is responsible for control of the KBE model component *Processes (P)*. The EMCp is focused on the *Processes (P)* to direct a development of enterprise output – products and services in the proper way (quality, time schedule, etc);
- *Knowledge management cycle (EMCz)* - this EMC of higher management level is implemented by component *Knowledge management functions (K)* responsible for the activities of KBE model component *Process management functions (F)*. The *EMCz* is focused on the alignment of *Business process (BP)* with the enterprise strategic goals.

As the content of the Process management cycle (EMCp) is adequate to formal description of the EMC in [12], next the particularities of the Knowledge management cycle (EMCz) are discussed next.

### 2.1. Analysis of the Knowledge Management Layer

By definition [12, 14] an Elementary management cycle (EMC) consists of the predefined sequence of the mandatory steps of information transformation: (*Interpretation (IN)*), *Information processing (IP)*), *Realization (RE)*); these steps compose a management cycle (an information feedback loop) of some controlled object.

The content of information and semantics of transformations performed by these steps of EMC depend on the subject area (particular domain of the enterprise). For instance, the subject area (the controlled object) of the Knowledge management cycle EMCz is a definite set of processes management functions (F). It is evident that the subject area of the EMCz (i.e. content of information and semantics of transformation performed by the EMCz) is totally different from that of the process management cycle EMCp. The EMCz deals with the information about the characteristics of management functions (quality, effectiveness, etc.), meanwhile the process management cycle

EMCp controls definite characteristics of products, services and state of a process (i.e. technological process).

So, the content (semantics) of information processed in these two management cycles (EMCz and EMCp) is unlike, different.

The steps of the EMCz (Interpretation (IN), Information processing (IP), and Realization (RE)) are defined as steps of KM activity focused on the re-engineering of the BP Management functions F on the BP management layer (Figure 1).

## 2.2. Decomposition of the Knowledge Management Layer

Knowledge management activity, as any other enterprise activity is arranged in a hierarchy, which can have infinite number of levels [15]. The first task is to identify finite number of design levels. According to MDA architecture four levels of modeling abstraction (details) are recommended [16]. On the basis of these considerations four hierarchically interrelated knowledge management levels (see Figure 2) are identified. Every higher-level component (K1, K2, K3, K4) of knowledge management domain is related with the lower level component (K2, K3, K4, BP) by some type of the Elementary management cycle (EMCz1, EMCz2, EMCz3, EMCz4).

Every component (K1, K2, K3, K4) of knowledge management domain is particular type of the knowledge management activity, it has definite (different) semantics. Each type of the Elementary management cycle (EMCz1, EMCz2, EMCz3, EMCz4) has a particular object of control and has a definite (different) semantics as well.

The *Knowledge management* (K) domain (Figure 1) of the KBE model is decomposed and four structural components are identified (Figure 2):

- The component K4 *Business process knowledge management* (KM level 4 - *Enterprise management level*). This component K4 consists of *Knowledge management functions* (F4), dedicated to control content of the Process management functions (F). K4 forms control attributes for Business process (BP) level. K4 uses interface S4 with the *Enterprise knowledge base* (KB) for the exchange knowledge about business processes (management functions (F) and Process (P)); K4 is related with BP by feedback loop EMCz4 (*Knowledge management control at the level 4*).
- Third component K3 *Enterprise knowledge management* (the KM level 3 - *Enterprise knowledge management level*). This component K3 consists of *Enterprise knowledge management functions* (F3), dedicated to control content of knowledge management functions (F4). This component K3 is aimed to complement knowledge of structural element K4 by using knowledge, stored in Enterprise knowledge base (KB) through interface S3; K3 is related with KM level K4 by feedback loop EMCz3 (*Knowledge management control at the level 3*).
- The component K2 *Knowledge base management* (the KM level 2 - *Enterprise-meta knowledge management level*). This component K2 consists of *Knowledge base management functions* (F2), dedicated to control content of Enterprise knowledge base (KB). This component K2 is aimed to improve the structure and content of the Enterprise knowledge base, defined in the *Enterprise meta-knowledge model*, i.e. to adjust the structure of KB with the business goals through the interface S2; Level K2 in turn is related with third KM K3 level by feedback loop EMCz2 (*Knowledge management control at the level 2*).



- Knowledge management is driven by organizational goals and objectives. In the highest KM level 1 (Enterprise Strategic management level) the component Business/IT strategic alignment (K1) defines strategic requirements for the Enterprise knowledge base meta-modeling. The component K1 consists of Knowledge Base meta-model management functions (F1), focused to control structure of Enterprise knowledge Base. Components K1 and K2 are interrelated by the feedback loop EMCz1 (knowledge management control at the level 1).

Summing up, four types of the Elementary management cycles (EMC), which have particular semantics, are identified by decomposition of Enterprise knowledge management (K) domain:

- EMCz1 – *Enterprise meta-knowledge management cycle*, focused on the alignment of the enterprise knowledge base content (i.e. enterprise meta-knowledge model) and business/ IT strategic goals;
- EMCz2 – *Enterprise knowledge management cycle*, aimed at the capturing knowledge for the enterprise knowledge base (required by the higher level component K1);
- EMCz3 – *Business process knowledge management cycle*, aimed at the adapting BP knowledge management functions;
- EMCz4 – *Business process management cycle* focused to the modification of the BP management functions.

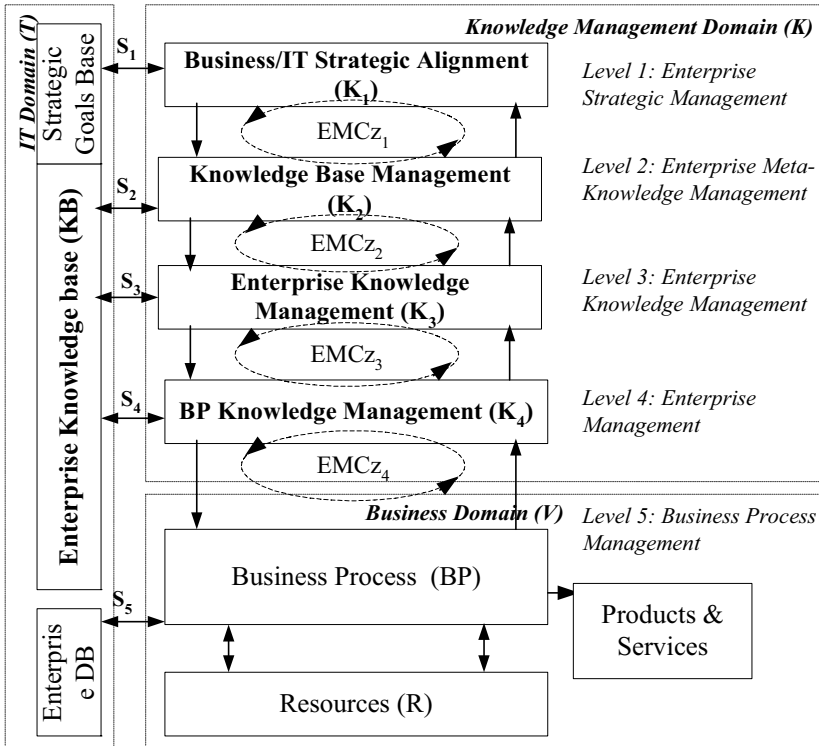


Figure 2. KBE model: knowledge management layers and knowledge management interactions

All structural elements of the KBE model are grouped into three interrelated enterprise domains: business domain (V), Knowledge management domain (K) and IT domain (T).

The Enterprise knowledge base (KB) contains integrated knowledge about all these enterprise domains as well as various relationships of these domains. In the KBE model relationships between IT domains (T) and Business (V) as well as Knowledge management domain (K) are defined as interfaces S1, S2, S3, S4, and S5. Relationship between Business domain (V) and Knowledge management domain (K) is represented as management control process EMCz4.

### 3. Interactions of the Enterprise Knowledge Management Components

According to J. M. Firestone, organizational Knowledge Management activity “is aimed at integrating the various organizational agents, components, and activities of the organizational knowledge management system into a planned, directed process producing, maintaining and enhancing an organization's knowledge base” [15]. The enterprise knowledge base along with its organizational and technological components constitutes Enterprise knowledge management system (KMS). A key aspect in defining the KMS is that both its components and interactions must be fully designed [15].

The semantics of structural components of the Elementary Management Cycles further is explained.

#### 3.1. Business Process Management Cycle (EMCz4)

The purpose of Business process management cycle (EMCz4) is development (generation) of particular knowledge to control an enterprise component Business process (BP). The component BP is comprised of a set of Management functions (F) and Process (P). The component BP is at Business domain (V), it is outside of the Knowledge management domain (K).

The semantics of steps of the Business process management cycle (EMCz4) (Figure 3) are as follows: IN4 – interpretation of some facts (characteristics) related with the controlled object – an activity BP; IP4 – processing of interpreted information (data, knowledge) and decision making (aimed to control an activity BP); RE4 – realization of decision (management control making, including transferring of manipulated variables (a particular decision) and influencing a controlled object – the component Business process (BP). The constraints on the Business process management cycle (EMCz4) are output of the component K3 (Enterprise knowledge management) and input of the interface S4 from Knowledge base (KB).

The activity IN4 performs an interpretation of the actual knowledge about the features (state) of Business process (BP). Characteristics (data and knowledge about a state) of Management Functions (F) and Process (P) are captured, transferred and conceptualized, using some criterions from the Enterprise Knowledge Base (EKB). This captured actual semantics of Business process (BP) is an input of the component BP knowledge management functions (K4). The activity IN4 comprises of a set of rules and procedures for transformation the actual data and knowledge about a state of Business process (BP).

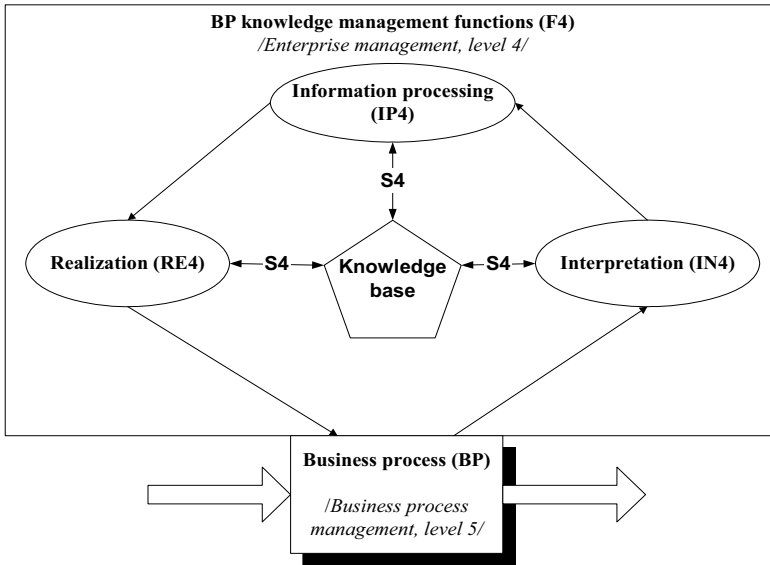


Figure 3. Business process management cycle (EMCz4)

The step IP4 – knowledge processing activity, aimed to define a set of manipulated variables – decision to control Business process (BP). The activity IP4 is a system of data and knowledge manipulation procedures focused for alignment of the content of component Business process (BP) (i.e. IP4 modify a list and logic of management functions F) in accordance with requirements of the higher level component K3 (these requirements are the output the step RE3 of the higher level management EMCz3) and actual knowledge accessed by interface S4 from the Enterprise knowledge base.

The step RE4 – the co-ordination activity, the feedback from higher-level knowledge management component K4 to business process management level 5. The RE4 is aimed to transfer manipulated variables (decision) and to influence the component Business process (BP), namely to modify Management functions (F).

### 3.2. Business Process Knowledge Management Cycle (EMCz3)

The management control EMCz3 is *knowledge adaptation cycle* focused to integrate the component BP knowledge management functions (K4) with the actual content of the Knowledge base (KB).

The purpose of Enterprise knowledge management cycle EMCz3 is development (generation) of particular new knowledge to develop Enterprise component BP knowledge management functions (K4), which is comprised of a set of Business Process management functions (F4).

The semantics of steps of the Business process knowledge management cycle (EMCz3) are as follows (Figure 4): IN3 – interpretation of facts (characteristics) related with the controlled object – an activity of the component K4, IP3 – processing of interpreted information (data, knowledge) and decision making (aimed to control the component K4), RE3 – realization of decision (management control making, including transferring of manipulated variables (a particular decision) and influencing a controlled object – the component K4).

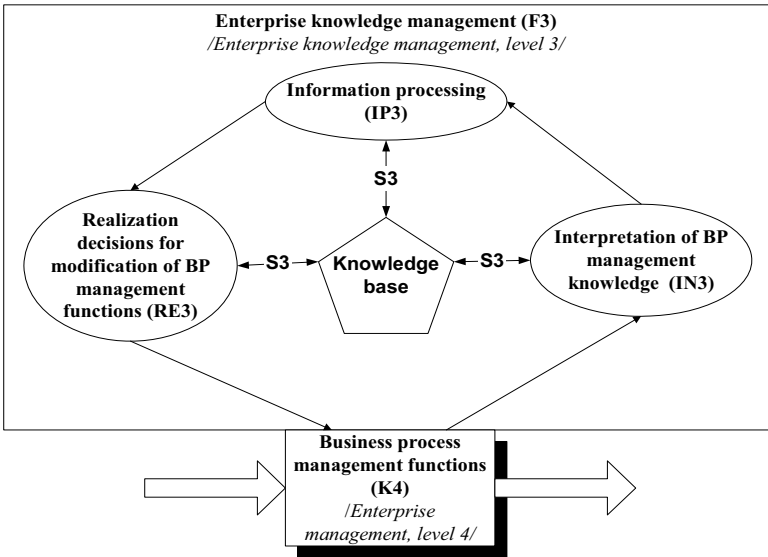


Figure 4. Business process knowledge management cycle (EMCz3)

The constraints on the Business process knowledge management cycle (EMCz3) are output of the component K2 (Knowledge base management) and input of the interface S3 from Knowledge base (KB).

The activity IN3 performs an interpretation of the actual knowledge about the features (state) BP knowledge management functions (F4).

The step IN3 comprises of a set of interpretation rules and procedures for transformation the actual data and knowledge about a state BP knowledge management functions (F4) for the integration with the step IP3. These transformations are aimed to fit the requirements of the IP3 – the next step of the enterprise knowledge management cycle EMCz3.

The step IP3 – knowledge processing activity, aimed to form a set of manipulated variables – decision to implement new features of the BP knowledge management functions. The IP3 is a system of data and knowledge manipulation procedures focused for modification of the content of component K4 with the requirements of the higher level component K2 (these requirements are the output the step RE2 of the higher level management EMCz2) and actual knowledge accessed by interface S3 from the Enterprise knowledge base.

The step RE3 – the co-ordination activity, the feedback from higher-level Knowledge management component K3 to Business process knowledge management functions (K4). The step RE3 is aimed to transfer manipulated variables (decision) and to influence the component K4, namely to modify Business process knowledge management functions (F4).

3.3. Enterprise Knowledge Management Cycle (EMCz2)

The management control EMCz2 is *higher-level knowledge adaptation cycle* aimed to modify the component Enterprise knowledge management (K3) of Knowledge management (K) domain.

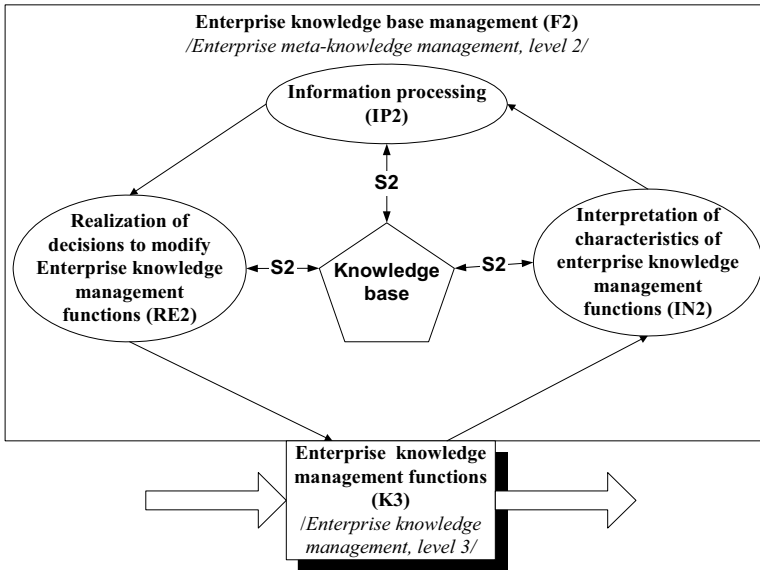


Figure 5. Enterprise knowledge management cycle (EMCz2)

The purpose of Enterprise knowledge management cycle EMCz2 is development (generation) of definite knowledge to adapt a set of Enterprise knowledge Management functions (F3) to the new requirements of the Knowledge base meta-model.

The semantics of steps of the Enterprise knowledge management cycle (EMCz2) are as follows (Figure 5):

IN2 – interpretation of facts (characteristics) related with the controlled object – an activity of the component K3;

IP2 – processing of interpreted information (data, knowledge) and decision making (aimed to control an activity of component K3);

RE2 – the step of realization (implementation) of decision aimed to influence a controlled object Enterprise management – the component K3.

The constraints on the Enterprise knowledge management cycle (EMCz2) are output of the component K1 (Business and IT strategic alignment) and input of interface S2 from Enterprise Knowledge base (KB).

The activity IN2 performs an interpretation of the actual knowledge about the features (state) Enterprise knowledge management functions (F3).

The step IN2 comprises of a set of interpretation rules and procedures for transformation the actual data and knowledge about a state of Enterprise knowledge management functions (F3) for the integration with the step IP2. These transformations are aimed to fit the requirements of the IP2 – the next step of the enterprise knowledge management cycle EMCz2.

The step IP2 – knowledge processing activity, aimed to form a set of manipulated variables – decision to modify the component Enterprise knowledge management (K3). The IP2 is a system of knowledge manipulation procedures focused for alignment of the content of component K3 with the requirements of the higher-level component K1 (these requirements are the output the step RE1 of the higher level management

EMCz1) and actual knowledge accessed by interface S2 from the Enterprise Knowledge Base.

The step RE2 is the co-ordination activity, the feedback from higher-level component Knowledge base management (K2) to the component Enterprise knowledge management (K3). The step RE2 is aimed to transfer manipulated variables (decision) and to influence the component K3, namely to modify Enterprise knowledge management functions (F3).

### 3.4. Enterprise Meta-Knowledge Management Cycle (EMCz1)

The management control EMCz1 is the *top level knowledge management activity* focused on the requirements for the scope and content of Enterprise knowledge base (meta-knowledge). The component Knowledge base management (K2) is responsible for the meta-knowledge management (at level 2 of Knowledge management domain.).

The purpose of Enterprise meta-knowledge management cycle EMCz1 is development (generation) of definite knowledge to modify the enterprise component Knowledge base management (K2), which is comprised of a set of Enterprise knowledge base management functions (F2).

The semantics of steps of the Enterprise meta-knowledge management cycle (EMCz1) are as follows (Figure 6):

IN1 – interpretation of facts (characteristics) related with the controlled object – an activity of the component K2;

IP1 – processing of interpreted information (data, knowledge) and decision making (aimed to change the knowledge base meta-structure);

RE1 – realization of decision (a strategic decision) aimed to influence a controlled object – the component K2.

The constraints on the Enterprise meta-knowledge management cycle (EMCz1) are input of the interface S1 from the knowledge base *Strategic Goals*.

The activity IN1 performs an interpretation of the actual knowledge about the features (state) enterprise knowledge base management functions (F2).

The step IN1 comprises of a set of interpretation rules and procedures for transformation the actual data and knowledge about a state enterprise knowledge base management functions (F2) for the integration with the step IP1. These transformations are aimed to fit the requirements of the IP1 – the next step of the enterprise knowledge management cycle EMCz1.

The step IP1 – a meta-knowledge processing activity, aimed to form a set of manipulated variables – decision to control the component Knowledge base management (K2). The IP1 is a system of knowledge manipulation procedures focused for alignment of the content of component K2 with the requirements of the actual knowledge accessed by interface S1 from the Enterprise Strategic Goals Base.

The step RE1 – the co-ordination activity, it is the feedback from higher level component Business and IT alignment (K1) to the component Enterprise knowledge base management (K3). The step RE1 is aimed to transfer manipulated variables (a strategic decision) and to influence the component K2, namely to modify Enterprise knowledge base meta-model and knowledge base management functions (F2).

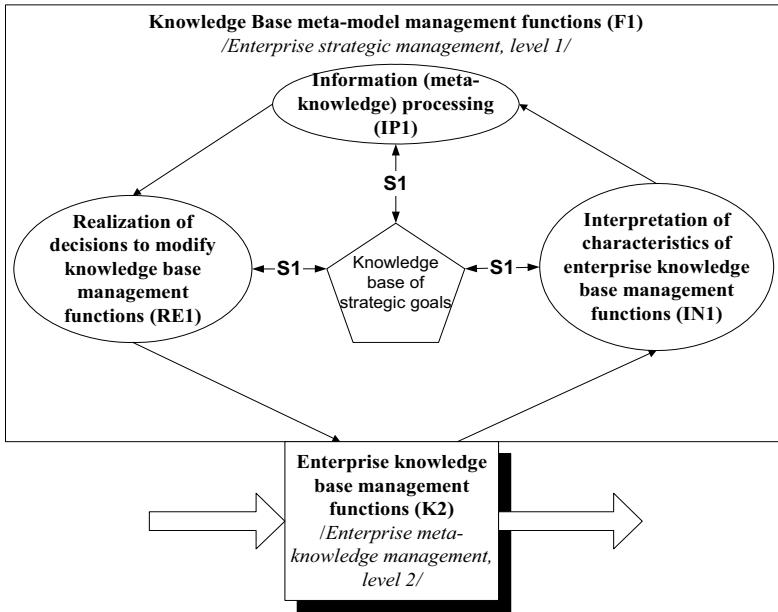


Figure 6. Enterprise meta-knowledge management cycle (EMCz1)

#### 4. The Major Knowledge Management Interfaces

The major knowledge management *Use Cases* and interfaces at the Enterprise knowledge management (K) domain are depicted at the *Use Case diagram* (UML) in the Figure 7. There are four types of *Actors* associated with particular level of the enterprise Knowledge management (K) domain: a top manager (a chief executive), a knowledge base administrator, an enterprise management expert, and a business process manager.

The enterprise knowledge self-organization activity is a responsibility of a chief executive. It includes interfaces with use cases Strategic knowledge management functions (F1) and Knowledge base management functions (F2).

Responsibilities of a knowledge base administrator includes interfaces for administration of use cases Strategic knowledge management functions (F1), Knowledge base management functions (F2), Enterprise knowledge management functions (F3), and BP knowledge management functions (F4).

The responsibilities of an enterprise management expert are focused on the development of a definite new knowledge and requirements for improvement of BP management functions (using interfaces with use cases Enterprise knowledge management functions (F3), BP knowledge management functions (F4), and Knowledge base management functions (F2)).

The BP managers access definite knowledge aimed to perform BP management and BP management control (using interfaces with the use case BP management functions (F)), and use interface with BP knowledge management functions (F4) to access definite knowledge for modification of BP management functions.

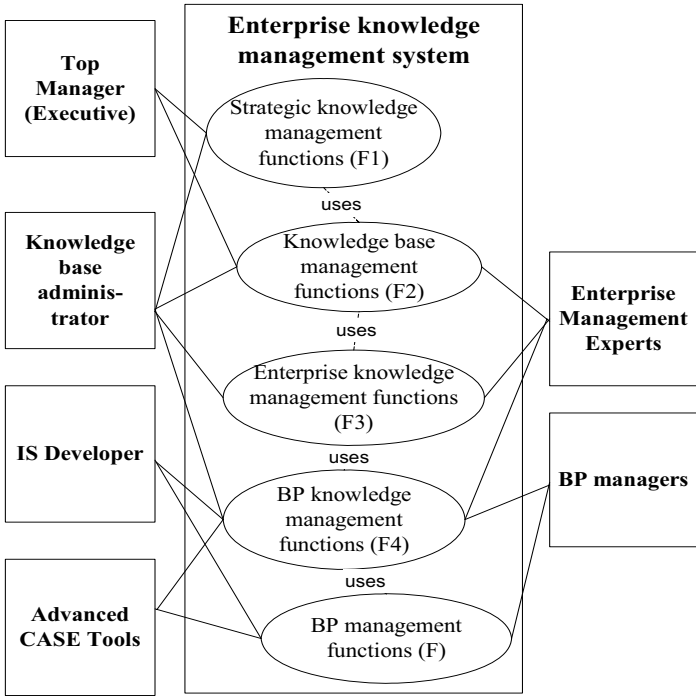


Figure 7. The major interfaces and use cases of Enterprise knowledge management system

5. Conclusions

The conception of the Knowledge-based enterprise embodies the vision of the more mature and more advanced enterprise from the knowledge management point of view and is the step towards intelligent enterprise systems. Advancement is seen here as the high formalization degree of the knowledge management activities, which results in the more efficient management and automation of business process and knowledge processes in the enterprise.

The Knowledge-Based Enterprise (KBE) model defines knowledge management activity in the enterprise in the formal way. KBEM is aimed for the development of the Enterprise knowledge base.

The formal description of the knowledge management activity is based on the concept Elementary management cycle (EMC). Business management control models deals only with the management in the business processes level (Elementary process management cycle EMCp). The peculiarity of the KBE model is that it reveals another - knowledge management level - and defines interactions between those two management levels of the enterprise, using the same EMC concept. The knowledge management layer of the KBE model contains a hierarchy of the knowledge management activities, defined as the particular types of the EMC.



All defined types of the EMC have their own semantics: the highest level of the knowledge management control is focused on the requirements for the scope and content of Enterprise knowledge base (meta-knowledge); the management controls of the second and third levels have to do with the enterprise knowledge adaptation; the lowest level knowledge management control is directed towards acquisition of particular knowledge to control an enterprise component Business process (BP).

Another important feature of the developed model is that the interactions (defined as interfaces S1...S5) between knowledge management domain and information technology domain (Enterprise Knowledge base) are defined formally too.

## References

- [1] Smialek, M., Balcerek, L. The knowledge-based content management application design methodology. *ICONS*, 2003. Available from: [www.icons.rodan.pl](http://www.icons.rodan.pl).
- [2] Porter, M. E. *Competitive Strategy: Creating and Sustaining Superior Performance*. New York: The Free Press, 1985.
- [3] Zack, M. H. Rethinking the knowledge-based organization. *Sloan Management Review*, vol. 44, no. 4, Summer 2003, pp. 67-71. Available from: <http://web.cba.neu.edu/~mzack/articles/kbo/kbo.htm>.
- [4] Gudas, S., Brundzaite, R. Decomposition of the enterprise knowledge management layer. In: *Proceedings of the Seventh International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2006)*, IEEE, 2006, pp. 41-47.
- [5] Gudas, S., Brundzaite, R. Enterprise knowledge modelling based on modified Value Chain Model. *Information Sciences*, vol. 35, 2005, Vilnius: Vilnius University Publishing House, pp. 179-192.
- [6] Gudas, S., Skersys, T., Lopata, A. Framework for knowledge-based IS engineering. In: *Advances in Information Systems ADVIS'2004*, Berlin: Springer-Verlag, 2004.
- [7] Kirikova, M., Bubenko, J. Enterprise modelling: improving the quality of requirements specifications. In: *Proceedings of the 17th IRIS Conference*, 1994, Finland, University of Oulu, pp. 939 – 953.
- [8] Loucopoulos, P., Kavakli, V. Enterprise knowledge management and conceptual modelling. In: P. P. Chen et al. (ed), *Conceptual Modeling, Current Issues and Future Directions*, LNCS 1565, Springer, 1999, pp 123-143. Available from: [http://www.aegean.gr/culturaltec/Kavakli/publications/pdf\\_files/er97\\_kavakli.pdf](http://www.aegean.gr/culturaltec/Kavakli/publications/pdf_files/er97_kavakli.pdf).
- [9] Strohmaier, M. B. *B-KIDE: a Framework and a Tool for Business Process Oriented Knowledge Infrastructure Development*. Shaker, 2005.
- [10] Mertins, K., Heisig, P., Vorbeck, J. *Knowledge Management – Concepts and Best Practices*. Berlin Heidelberg New York: Springer Verlag, 2003.
- [11] Maier, R. *Knowledge Management Systems: Information and Communication Technologies for Knowledge Management*. Springer, 2004.
- [12] Gudas S., Lopata A., Skersys T. Approach to enterprise modelling for information systems engineering. *Informatica*, vol. 16, no. 2, 2005, pp. 175-192.
- [13] Kampfner, R. R. Modeling the information-processing aspect of organizational functions. In: *IEEE Conference and Workshop on Engineering of Computer-Based Systems*, 1999, p. 75.
- [14] Gudas, S. A framework for research of information processing hierarchy in enterprise. *Mathematics and Computers in Simulation*, no. 33, 1991, pp. 281-285.
- [15] Firestone, J. M. Enterprise Knowledge Management Modeling and Distributed Knowledge Management Systems. Executive Information Systems, Inc. 1999. Available from: <http://www.dkms.com/papers/ekmdkms.pdf>.
- [16] Hettinger, M. K. Model-driven architecture, processes and methodology from the perspective of the “modeling discipline”. In: *MDA™ Implementers' Workshop: Succeeding with Model Driven Systems*, Orlando, Florida, 2003. Available from: [www.omg.org/news/meetings/workshops/MDA\\_Wksp\\_Manual/00-6\\_T2\\_Hettinger.pdf](http://www.omg.org/news/meetings/workshops/MDA_Wksp_Manual/00-6_T2_Hettinger.pdf).

This page intentionally left blank

# Databases

This page intentionally left blank

# Methods for a Synchronised Evolution of Databases and Associated Ontologies

Andreas KUPFER<sup>1</sup>, Silke ECKSTEIN, Britta STÖRMANN, Karl NEUMANN and  
Brigitte MATHIAK

*Institute of Information Systems, TU Braunschweig, Germany*

**Abstract.** Ontologies are well applicable for the description of scientific databases and thus provide appropriate means to link different databases. In a first step it is useful to generate initial ontologies from the respective database schemas. They can be completed with more specific information later on. If the database schemas change, however, these changes should now automatically be propagated to the respective ontologies. Using a realistic example from the biological field of signal transduction pathways, we demonstrate that the suggested approach is both, feasible and reasonable.

**Keywords.** OWL ontology, database schema, evolution

## Introduction

In biological research, many databases were created over the last few years. They store a rapidly growing amount of information about topics like protein sequences, genomes, organisms or enzyme classification. Currently, knowledge from biological research is stored in over 850 databases, counting only public accessible ones. The number of biological research databases is increasing every year, and has increased in January 2006 from 719 to 858 [1]. Almost all of them contain information that can be combined with the information in other databases [2].

Connecting these diverse data sources is a challenging task which can be supported by ontologies describing them [3]. For example, an ontology can express that two databases, like PRODORIC [4] and TRANSPATH [5]<sup>2</sup>, both contain information about molecules in their specific context. It can also include the information where content about this topic is stored in the database. This is more precise than just using database catalogue information. Since the context can be expressed in a machine-readable form, a program integrating such databases could decide if the database is about molecules in a specific organism, e.g. human, mouse or yeast.

So far, ontologies in biology are mostly used as a controlled vocabulary to cope with the ambiguity or heterogeneity of biological data. Even large controlled vocabularies, like GeneOntology [6], only apply to a fraction of the available data. The vocabularies

---

<sup>1</sup>Corresponding Author: Andreas Kupfer, Technical University of Braunschweig, Institute of Information Systems, Mühlenpfordtstraße 23, 38106 Braunschweig, Germany; E-mail: kupfer@ifis.cs.tu-bs.de.

<sup>2</sup><http://www.biobase.de/pages/products/databases.html>

can be used as a source of defined concepts when describing a database. Other sources can be the conceptual model from the database design or foreign key constraints in the database schema. It is also possible to build a specialized domain model, as we did for this paper.

A database schema often changes over time especially in databases used for research [7]. Unfortunately, an ontology for a specific database is only usable as long as it matches to the database schema and maintenance of these changes is time-consuming manual work. Therefore it is worthwhile to reduce the required work by automating tasks.

Before we discuss our method to synchronise the database schema evolution and the ontology evolution, we start by creating a database ontology from a given database schema. Our database schema used is based on a well known biological research database. The created database ontology provides links to all tables and attributes, allowing each of them to be annotated with semantical concepts and associations. For our example, we create a domain model and connect it to the database schema. The database schema is allowed to change, so we explore how to transfer those changes to the ontology simultaneously and keep both versions attached to each other.

In the following section we give an overview over the related work. Section 2 explains in more detail how we produce the initial ontology from a database schema and introduces the schema of our example database. Section 3 covers the design of a signal transduction pathway ontology, which we use to annotate the schema. Section 4 goes into detail on how annotations can be made on the ontology in order to store semantics. In section 5 all possible changes of the database ontology are discussed and we show, how they can be reproduced in the annotated ontology. We conclude with some remarks about our future work.

## 1. Related Work

In the semantic web ontologies are used to allow queries on inhomogeneous data sources [8,9]. By making the semantics of metadata machine-interpretable, ontologies offer the possibility to deal with complex queries, which require an uncovering of semantic interrelations [10]. Various languages were developed to describe ontologies, but since 2004 there is a promising W3C standard available named Web Ontology Language (OWL) [11]. OWL has three sub-languages: OWL-lite, OWL-description logics (DL) and OWL-full, which differ in their complexity. OWL-DL is OWL-full with additional restrictions and OWL-lite is OWL-DL with less language elements [12].

An ontology can describe a database in greater detail than a data type classification from a database schema could do, because it does not only describe syntax but semantics as well. Existing databases and information systems, even if they are belonging to the same area of application, often contain relations or classes which are ambiguous or heterogeneous. Supplementing controlled vocabularies with ontologies describing database schemas provides a more abstract view to the information space [13].

Creating an ontology resembles a conceptual specification of a database. Differences exist in goals, expressiveness and extensibility [14]. Various methods and methodologies for creating ontologies exist, details of this process can be found in [15]. Often, ontologies are created for already existing data. Most approaches related to automatic ontology generation use large text corpora as input [16,17]. There are also concepts for generat-

ing ontologies which reflect the structure of XML documents [18], which resembles our input data of database schemas most.

To demonstrate the proposed methods, we use the TRANSPATH [5] database. Based on information gained by profound validation of biological literature this database contains essential data for the generation of signal transduction pathways, and whole gene regulatory networks respectively. Additionally the database provides means to integrate data obtained by microarray analysis into the context of existing signal networks. In [19] it has been suggested to build an ontology with the domain concepts of signal transduction pathways. We will create one in section 3 for annotation purposes.

We will not examine the evolution of database schemas itself, as we assume that database systems are capable of applying any schema changing SQL statement, which reflect all database evolution constructs regarding relational database systems [20]. Regarding automatic maintenance of ontologies, manual expert support is still needed [21]. But in the meantime there exist methods which can detect changes in ontologies automatically and correct problems caused by these changes [12]. Algorithms for generating a mapping between ontologies are available as well [22].

## 2. Schema-to-Ontology Mapping

This section describes an abstract database ontology and the automatic generation of database ontologies, which represent given database schemas. We start by designing a general ontology for database schemas. This can be used to map each database schema with all tables and data type information to an individual ontology in OWL-lite format. We demonstrate this procedure on a database, derived from TRANSPATH. For clarity, we use the term 'table' for the relations of a database and 'relation' for the relationships between concepts of an ontology.

### 2.1. The Database Ontology

The abstract database ontology has concepts for the terms *Database*, *Table*, *Attribute*, and an object property *consistsOf* to create a hierarchy between them. In this ontology a database consists of several tables which consist of several attributes (columns). Each table is identified by its unique name and each attribute additionally by an XML schema data type. Table and column names are stored as an ID of the respective instance. These structures are similar to those in [23].

Identifiers of instances in OWL are encouraged to be document wide unique. This is not true in databases where columns in different tables can have the same name. To cope with this, we add the table name to each attribute as its suffix. The original name is stored as a label of the instance, so editors can show the usual attribute name. At this time, special data types, primary keys, cardinalities and integrity constraints are not needed and hence not transformed to the ontology. They can be extended later on though. In principal, foreign keys represent associations between tables and may be expressed as semantic relations. But modelling details, for example aggregation or composition can not be distinguished any more. Instead, a generic object property expressing all foreign key relations may be defined.

An OWL ontology for our abstract database is created by declaring these three concepts and semantic relations (cf. sec. 4). The object property *consistsOf* will be used to

create a hierarchy between the defined concepts. Since we use OWL-lite, there are no restrictions, to enforce the structure of our hierarchy in the instances. Likewise, there are no sequence restrictions. The order of attributes in a table has to be determined by their position in the document. These restrictions are not needed, though, because ontologies per se and our specific application do not demand them.

## 2.2. Recreating a Signal Transduction Pathway Database from the TRANSPATH Export

The TRANSPATH database can be downloaded both as XML files and as flat files to be used with cgi-scripts. To create realistic examples, we create a generic signal transduction database (abbreviated as GiST below) by reimporting data from the TRANSPATH in a POSTGERSQL database system.

In the current release version 7.2 the XML export consists of 6 files — molecules, reactions, pathways, genes, annotations and references — with a total size of about 340MB. All files are datacentric XML documents and they are described by a common DTD (Document Type Definition) document. The data import is done using the java tool xml-dbms<sup>3</sup> version 2.0α3, which uses an object-relational method [24].

Using the tool to get a database schema from the DTD, it produces 88 tables. Initially, most of the foreign keys are lost, since they were not exported as IDREFs, but as xlink references instead. We decided to replace all xlinks with equivalent IDREFs. Also, all empty values were deleted, which had been exported as whitespaces. The new database schema consists of 63 tables and applicable foreign key constraints. Below is the generated schema for one table:

```

1 CREATE TABLE "GENE" (
2   "UPDATOR"    VARCHAR(255) NULL,   "NETWORKFK" INTEGER NOT NULL,
3   "FULLNAME"   VARCHAR(255) NULL,   "GENEPK"      INTEGER NOT NULL,
4   "SPECIES"    VARCHAR(255) NULL,   "ID"          VARCHAR(255) NOT NULL,
5   "CREATOR"    VARCHAR(255) NULL,   "EXTID"       VARCHAR(255) NULL,
6   "NAME"       VARCHAR(255) NOT NULL,
7   "SECID"      VARCHAR(255) NULL,
8   CONSTRAINT PRIMARYKEY PRIMARY KEY( "GENEPK" ),
9   CONSTRAINT NETWORKFK FOREIGN KEY( "NETWORKFK" )
                                REFERENCES "NETWORK" ( "NETWORKPK" );

```

This database schema still has some flaws. The data types are generic because the data types in a DTD document can not be more specific. The conversion process has added new primary keys instead of using the existing IDs (lines 3, 4 on the right hand side). However, since this schema can be attained with minimal user input, we use it for the examples in the following sections.

## 2.3. Generating Ontologies

To generate a database ontology, we use the extracted database schema from a given database as input. This information is then stored as instance data of the abstract database ontology from section 2.1. We have implemented this process as a java tool using the XML library XOM to store the generated OWL ontology.

<sup>3</sup><http://www.rpbouret.com/xmldbms/>



The database schema is stored in the system tables of the database and can be read by standard SQL queries. To cope with non-compliant database systems we will implement wrappers which translate schema queries to the corresponding database systems. To cover most database systems with one wrapper implemented so far, we parse SQL statements from a database dump to be used as input for our ontology generation program. The schema dump can be obtained from various database management systems, e.g. in PostgreSQL with `pg_dump`.

To outline our mapping process, we use the database schema of the GiST database from the last section as an example. The program starts by creating an instance of *Database* and then it parses each SQL statement from the given database schema dump. An instance of the concept *Table* is created for each occurring table and an instance of the concept *Attribute* for each column. The data type of each attribute is stored as a reference to an XML Schema data type. The data types from the schema are not mapped exactly as predefined data types in XML schema are more generic, e.g. any char, varchar or text data type is mapped to an XML Schema string.

The hierarchy structure between the instances is created by connecting them with the object property *consistsOf* according to the abstract database model. An example of the generated instances for the table *Gene* is shown below:

```
<Database
  xmlns="http://www.ifis.cs.tu-bs.de/htmlXd/home/kupfer/dbOnt#">
  <consistsOf>
    <Table rdf:ID="GENE">
      <consistsOf>
        <Attribute rdf:ID="UPDATOR__GENE">
          <rdfs:label
            rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
            >UPDATOR</rdfs:label>
          <consistsOf
            rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </Attribute>
        </consistsOf>
      <consistsOf>
        <Attribute rdf:ID="NETWORKFK__GENE">
          <rdfs:label ...>NETWORKFK</rdfs:label>
          <consistsOf
            rdf:resource="http://www.w3.org/2001/XMLSchema#integer" />
          </Attribute>
        </consistsOf>
      </Table>
    </consistsOf>
  </consistsOf>
  ...
</Database>
```

### 3. An Ontology for Signal Transduction Pathways

Next, this database ontology can be annotated with semantic concepts and associations. These annotations are necessary prerequisites for data integration and semantic query. To provide realistic data for the annotation process, we create a signal transduction pathway ontology for the GiST database. Although this ontology is certainly inspired by the

database we used, we aim for an independent representation of the concepts involved. In an ontology, we have to take in account the Open World Assumption. The goal is to have an precise and open definition of the concept and their relations. While databases seek to represent the data, ontologies focus on the concepts. In section 4, we will discuss how this ontology can be implemented as part of the database ontology generated in the last chapter.

As you can see in Figure 1, the base for the model is a classic *isPartOf* hierarchy of the pathway, its chains and the reactions occurring in the chains. While pathway and chain are concepts defined well enough, reaction has to be clarified before being used in an ontology, to avoid synonym definitions. A reaction (in the chemical sense) includes the chemical modification of a molecule. In signal transduction pathways, however, there are also molecular interactions taking place without actual chemical modifications. For example, molecules bind to each other to form a new complex, with new chemical functions. Therefore we chose the term *SignalTransductionStep* for these processes and assigned two subclasses, *Reaction* and *Interaction*, to cover these differences and to clear up the semantics involved.

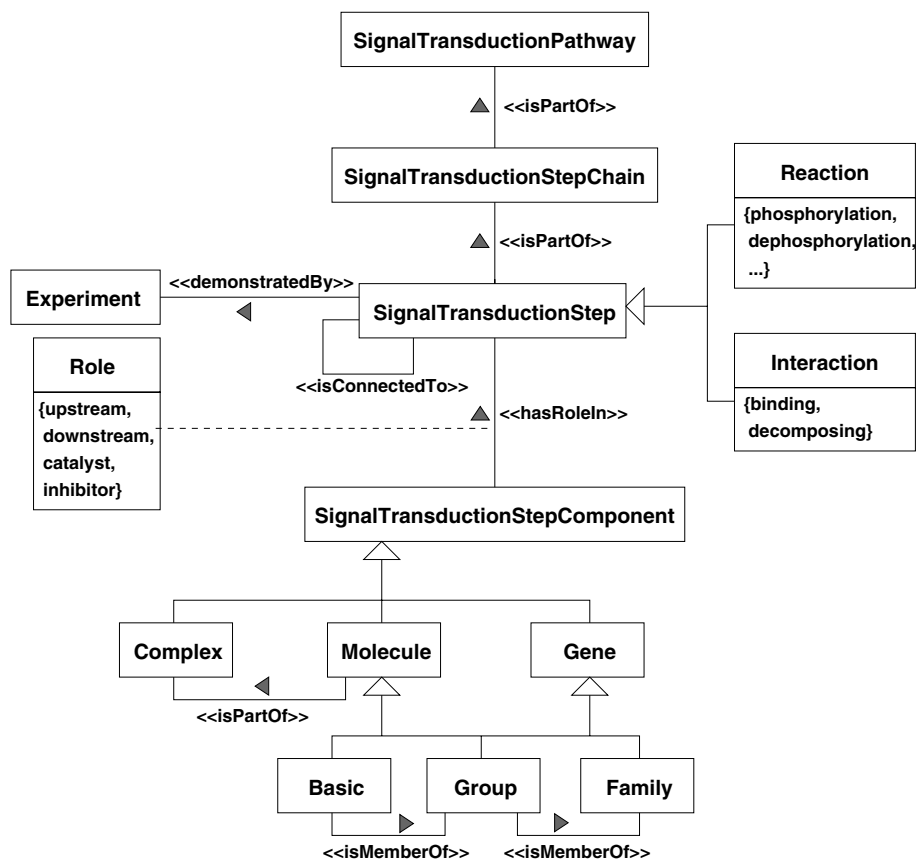


Figure 1. UML class diagram for the signal transduction pathway ontology

When examining the information value of the concepts, *SignalTransductionPathway* and *SignalTransductionStep* (step for short) can be easily identified as structural units. A *SignalTransductionStepChain* includes a number of steps that match each other, i.e. some product of the first step is involved in the next step, etc. So, the chain defines structure in the pathway. While the steps themselves may be matched by comparing reactant and product, the network resulting from that comparison is often large, complex and unordered. Chains are often defined in literature and may thus serve as a guide through the pathway.

A *SignalTransductionStep* contains more information yet unmentioned, from two distinct categories. First, there are the chemical features of a reaction. Even though it is already divided between reaction and interaction, the subclass *Reaction* could be specified more precisely by distinguishing the chemical modifications, like e.g. phosphorylation vs. dephosphorylation. Second, there is meta information concerning the experiment, which demonstrated the step.

Another very prominent concept in signal transduction pathways are the participants of the reactions and interactions. While they have a distinct classification taxonomy on their own, they are also connected to *SignalTransductionStep* in several roles. Looking at the chemical formulae of the steps, the participants of those reactions or interactions can either be basic molecules, belong to a family or group of molecules, be bound in a complex or represent a gene that is activated or inhibited. Integrating this information into our model, there is an *isMemberOf* hierarchy between basic, group and family. Also, a complex can be broken down into molecules and is connected to them via an *isPartOf* relationship. The semantical difference between the two relations *isMemberOf* and *isPartOf* is subtle, but can be clearly defined. A part is vital to the whole, changing the whole's identity when missing. A pathway with missing reactions is either incomplete (broken) or a new sub-pathway. A complex missing one of its components might behave in a different way, e.g. by conveying signals differently. In an *isMemberOf* relationship the member is not defining to the whole. A group of molecules is not defined by its members, but by their similarity. If a molecule should be taken out from the list, the similarity still holds.

While all three participants of reactions/interactions (i.e. molecules, complexes and genes), can take part in reactions or interactions, they can be clearly distinguished by several chemical and other properties. To represent their relation to *SignalTransductionStep*, an abstract concept was introduced called *SignalTransductionStepComponent* to which they are all subclasses. This concept serves like a role the components can play in the context of *SignalTransductionPathways*. These roles can be differentiated further. The component might be upstream, going into the reaction/interaction and being changed/bound there. Downstream are components that appear as a result of a step. Additionally, components can be catalysts only mediating a reaction, but not being altered by it. Other components are known to inhibit a step when active. It may seem counter-intuitive to include inhibitors as components as they interfere negatively with a step, preventing its execution. Still, in the context of signal transduction, an inhibiting step transports as much information as an activating step.

## 4. Connecting Semantic Information

Once the database ontology for a specific database is generated, semantic information about the application domain can be added. In this process an ontology editor is used to add or import new concepts from the domain ontology and connect them via relations to the instances of the database ontology. In this section we outline the annotation process by using the database ontology for the GiST database (from sec. 2.3), supplementing it with the signal transduction pathway ontology from section 3.

### 4.1. Adding Concepts

When starting the annotation process, the generated ontology only contains the concepts to describe a relational database schema (cf. sec. 2.1). It can be further enriched by adding new concepts describing the domain and relating them to the generated instances from the database.

From Figure 1 we can identify several concepts, which can be added to the database ontology by an editor like Protégé. As an example, the declaration for *SignalTransductionStepChain* (line 1) is shown below. Concepts should have a human readable comment (lines 3-4) to describe it too. These concepts can now be used to annotate the database schema.

```

1 <owl:Class rdf:ID="SignalTransductionStepChain">
2   <rdfs:subClassOf rdf:resource="#TopClass"/>
3   <rdfs:comment
4     >Assembly of SignalTransductionSteps with inherent order. The
      order is maintained by the sequential production or modification
      of the participating SignalTransductionStepComponents in a step,
      each of the steps in a SignalTransductionStepChain thus being
      prerequisite for the next one. SignalTransductionStepChains are
      found in literature as substructures of whole SignalTransduction
      Pathways and represent single alternative threads after
      furcation of SignalTransductionPathways.</rdfs:comment>
5   <isPartOf rdf:resource="#SignalTransductionPathway"/>
6 </owl:Class>

```

As an alternative to store our concept definitions in the database ontology, they can be stored as an external domain ontology and referenced like in section 4.3 below. This would allow to reuse concepts to annotate another database ontology about another signal transduction pathway database.

### 4.2. Adding Object Properties

To use the new concepts or to express associations between instances of the database schema, we add relations or object properties, as they are named in OWL. Additional relations must be declared once and can be used as instances then.

Different sources contain information about relations in the database. If the conceptual model is available, it can be used as a starting point for the manual annotation process. UML class diagrams often contain named associations, that were lost during the

transformation from a conceptual model to a relational model. These associations can be used as relations between concepts of the database ontology. Alternatively, the foreign key constraints in the database schema can be used, but they lack a semantical description. As a third option, we can design our own domain model (cf. section 3) and create the relations used in it.

To illustrate the process we start by creating a relation to connect semantical concepts with specific content of the database. Using an ontology editor will add code to the ontology (lines 1-8 below) when we declare the relation *containsDataAbout*. We can also define the inverse relation *hasDataIn* (lines 9-14).

```

1 <owl:ObjectProperty rdf:ID="containsDataAbout">
2   <rdfs:comment
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
3   >connecting a specific table or attribute to a concept
   </rdfs:comment>
4   <owl:inverseOf>
5     <owl:ObjectProperty rdf:ID="hasDataIn"/>
6   </owl:inverseOf>
7   <rdfs:domain rdf:resource="#TopClass"/>
8 </owl:ObjectProperty>
9 <owl:ObjectProperty rdf:about="#hasDataIn">
10  <rdfs:comment
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
11  >connecting a concept to a point in the database schema,
      where such data is stored</rdfs:comment>
12  <owl:inverseOf rdf:resource="#containsDataAbout"/>
13  <rdfs:domain rdf:resource="#TopClass"/>
14</owl:ObjectProperty>

```

Then an instance of the new defined relation is created for each pair of concepts or instances connected. The example below shows, how a new relation *containsDataAbout* (line 2) is used. This instance of the relation is originating from the table *Chains* instance, as it is added inside its declaration, and relates the table to the concept of a signal transduction chain, defined in section 4.1. Relations can originate from or target at any concept or instance of concepts in our database ontology, but not other relations.

```

1 <Table rdf:ID="CHAINS">
2   <containsDataAbout
      rdf:resource="#SignalTransductionStepChain"/>
3   <consistsOf>
...

```

The logical model from the database schema differs from the conceptual model we use for semantic annotation. For example, an n-m association or normalisation results in additional tables. In the annotation process such instances of *Table* should be ignored and the originating concepts should be associated by new object properties.

#### 4.3. Import from other Ontologies

The key for data integration is to relate to concepts, which are widely adopted. So, instead of creating our own concepts, we may also link to existing concepts from other ontologies

to our database ontology. In OWL, we can easily use URIs to refer to remote resources like other ontologies. Domains like biology or knowledge management already have ontologies to relate to, so we may simply link to them using URIs.

With the statement `owl:equivalentClass` we can define that classes from different ontologies describe the same concept. To link our class of a signal transduction pathway to an existing one we can easily add it in the declaration:

```
1 <owl:Class rdf:ID="SignalTransductionPathway">
2   <hasDataIn rdf:resource="#PATHWAY"/>
3   <owl:equivalentClass
4     rdf:resource="http://www.geneontology.org/owl/#GO_0007242"/>
5   ...
6 </owl:Class>
```

By adding line 3 in the example above we have linked our concept to a similar concept in an external ontology. The Gene Ontology concept no. 7242 is an intracellular signalling cascade, in context of signal transduction (no. 7165). A reasoner can now use additional semantic information which is defined in the other ontology. The use of common concepts in different ontologies is a key process in data integration.

External ontologies can change as well, so the referenced concept might not exist anymore. For our database ontologies, we use CVS and place the version information string in the element `owl:versionInfo` of each ontology accordingly. This allows to jump to past versions until the deleted definition can be found. A system like this might be applicable to external ontologies as well.

## 5. Synchronisation on Database Schema Evolution

When the ontology is automatically produced and manually enriched, we have to consider the database evolution. The ontology is a stand-alone file which would become outdated, if the database schema changes. In this section we will consider all possible schema changes which affect the database ontology.

We assume that the evolution of the database schema is stored in a log file that contains all SQL commands given to the database manipulating the schema. For each of the schema manipulation commands we have implemented a special routine to adjust the changes in the corresponding ontology.

SQL statements can change specific syntactical parts of the database schema. Possible schema changes are creation, deletion and alteration of tables or columns. These changes can be transferred to the related database ontology by our system. The system can not decide, if the user had a change of the semantical concept in mind. Therefore the process is semi-automatical. We will discuss the possible schema changes and their implication on the database ontology in the following subsections.

### 5.1. Creating

The most obvious change is a new table, added by a `CREATE TABLE` command. We use the already implemented methods to create new instances of the concept *Table* for each new table as discussed in section 2.3. `ALTER TABLE ADD COLUMN` works similarly with respect to attributes. We create a new instance for the new attribute and insert it in the corresponding table.

## 5.2. Deleting

When a column or a table of the database schema is removed, the corresponding instance in our database ontology has to be removed, too. In order to delete something, we first have to find it. The identifier of the original instance is computed and then it is searched for in the ontology. In case of a table, its identifier is equal to its name. If it is an attribute, the identifier is equal to its name and the name of the table. Also, we have to find all references (like `rdf:resource`, cf. section 2.3) which link to the deleted one, as not to leave broken links. After that, the instance can be removed as well as all related annotations.

The question that remains is, if we really want to delete everything about the old concept. In our analysis of real life occurrences of deletions in database schemas, it seems as if there are in fact two reasons or semantical concepts that lead to a deletion. It may be a removal of a not-used concept, like an attribute that seemed useful at the time of table creation, but is not useful anymore. Or it may be a replacement of an outdated concept with a new one, like in a typical deletion-creation pair. In both cases it is not useful to keep the old instance, since the instance will not be developed further. Whether or not to delete the references is a completely different issue. In case the deletion is just part of a hidden replacement, the semantical annotations should be preserved so they can be easily re-attached to the new upcoming concepts. The user has to decide if the references should be deleted. The instance will be deleted and if the references would not be used in another instance they would remain as broken links.

## 5.3. Altering

With the SQL statement `ALTER TABLE` various changes are possible on the database schema. They can be classified in three categories: There are changes which do not affect any part of the database ontology, e.g. changes of constraints, views or storage related changes. These are discarded. There are 2 categories left which are transferred to the ontology: renaming and data type changes.

Tables and columns can be renamed. The content of the database will not be changed by this operation. Although it may seem tempting to assume that a name change just means that the name changed and the actual concept stayed the same, this may not necessarily be true. An example is shown below:

```
ALTER TABLE GENE RENAME COLUMN UPDATOR TO LastUpdater;
```

We rename in the table *Gene* the column *Updater* to *LastUpdater*. With this information alone we can not decide if it is the same for all annotated concepts. If this attribute only contains a relation to a concept like *Person* it would not matter. But there is a semantic difference between any updater and the last updater, which might have been annotated. So we have to offer the option of either deleting all involved annotations or not to the user in any case.

Since attributes are concepts by their own right, all alterations are treated as implicit deletions and creations. The decision that remains, is whether it contains the same concept as before. In the case of changing an existing instance, we first have to find the instance (like described in section 5.2) and then make the needed changes. If a column is renamed we generate a new identifier based on its new name and replace it in the instance

of attribute. If we change the name of a table we also have to change all identifiers of its attributes, because the name is used there as well.

According to the user decision we either delete the annotations or not. Because we use the names of tables and columns in our identifiers we have to act in both cases. When we change the identifier of an instance, it would retain all relations which relate from the instance but loose all relation which relate to this instance. Therefore, if we want to delete the annotations we delete all additional elements from the instance and delete all relations in the ontology which target the old identifier. In this example, any relation in *Updater* (like *containsDataAbout*) and any relation targeting this instance is affected and either updated or deleted. We can only check links in the current ontology. To cope with references from external ontologies we create a link with `owl:sameAs` to point external references to the new identifier.

The data type of a column can be changed. Transferring this kind of change to the database ontology is quite easy. For example, changing the data types of the primary keys according to the data:

```
ALTER TABLE GENE ALTER COLUMN ID TYPE char(7);
```

In fact, we just compute the new corresponding XML schema data type, which would be a string again, and replace the old type declaration in the specified instance of attribute. The data types in the database ontology are quite generic, so there will be no change in this case.

## 6. Conclusions

In this paper we showed how ontologies and relational database schemas can be evolved in parallel. We motivated our approach, presented an automatic mapping from database schemas to database ontologies. As a complex example from the real world, we designed an ontology about the domain of biological signal transduction pathways and used data from the TRANSPATH database. Then we discussed the evolution of the database ontology by adding semantic context and finally illustrated the process in which changes of the database schema can be applied to the related database ontology.

At the moment the logging of schema changes and the ontology generation process is implemented with schema dumps in text format as input. Other tasks like the schema extraction for different database systems and tool support in schema annotation, will be implemented in near future. Later on, we plan to extend the framework towards processing of OWL-DL ontologies allowing us to map integrity constraints from the database schema to restrictions in the ontology. There is also more information in the database schema available like views, which could be added to the abstract database ontology and exported to the generated database ontologies as well. So far, we concentrated on relational databases, because they are most common, but we are confident that similar approaches can be made to other database models, too.

The inverse direction, creating a database schema from the database ontology is possible. This would allow to use only one model, which describes syntax as well as semantics. In this case no information would be lost during the database design. Since we are mostly looking into already existing databases, we have not covered this problem



yet. Still, it would be an interesting approach to design a database from scratch in that way.

Thinking beyond the scope of this project we would like to add semantic query processing for the ontologies [25]. This would allow users to easily find data without knowledge of the database schema in a database. The user would enter a query, like "Which publications are available on the protein X and the cell type Y?", using the concepts of the domain. It would be converted to or written in an ontology query language, e.g. SPARQL, asking for instances which *containsDataAbout* the concepts the user is interested in. Then this query could be interpreted by a reasoner on the database ontology. From the result, we will be able to deduce the involved tables and attributes of the database and issue a SQL query to it. So, the requested data from the database could be fetched without an explicit knowledge of the database schema by the user. By merging the database ontologies, which is very simple, the same query could be answered and returning results across all databases which use these database ontologies.

## Acknowledgments

The first author was funded by the German Federal Ministry for Education and Research, BMBF, Grant No. 031U210C as a member of the Interogenomics Center (<http://www.intergenomics.de>).

## References

- [1] M.Y. Galperin: The Molecular Biology Database Collection: 2006 update Nucleic Acids Research **34** (2006), D3–5.
- [2] Z. Lacroix and T. Critchlow: Bioinformatics – Managing Scientific Data, Morgan Kaufmann, 2003.
- [3] J.J. Lu and C.N. Hsu: Query answering using ontologies in agent-based resource sharing environment for biological web information integrating, In S. Kambhampati and C.A. Knoblock, eds.: Proc. of IJCAI-03 Workshop on Inf. Integr. on the Web. (2003), 197–202.
- [4] R. Münch, K. Hiller, H. Barg, D. Heldt, S. Linz, E. Wingender and D. Jahn: PRODORIC: prokaryotic database of gene regulation, Nucleic Acids Research **31** (2003), 266–269.
- [5] M. Krull, S. Pistor, N. Voss, A. Kel, I. Reuter, D. Kronenberg, H. Michael, K. Schwarzer, A. Potapov, C. Choi, O. Kel-Margoulis and E. Wingender: Transpath®: an information resource for storing and visualizing signaling pathways and their pathological aberrations, Nucleic Acids Research **34** (2006), D546.
- [6] M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, M. Harris, D. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. Matese, J. Richardson, M. Ringwald, G. Rubin and G. Sherlock: Gene Ontology: tool for the unification of biology, Nature Genetics **25** (2000), 25–29.
- [7] H. Fan and A. Poulouassilis: Schema evolution in data warehousing environments - a schema transformation-based approach, In P. Atzeni, W.W. Chu, H. Lu, S. Zhou and T.W. Ling, eds.: ER 2004, 23rd Int. Conf. on Conceptual Modeling, Proc. LNCS 3288, Springer Verlag (2004), 639–653.
- [8] P. Bouquet, G.M. Kuper, M. Scoz and S. Zanobini: Asking and Answering Semantic Queries, In: Meaning Coordination and Negotiation Workshop (MCNW-04) in conjunction with Int. Semantic Web Conf (ISWC-04), 2004.
- [9] H. Stuckenschmidt and F. v. Harmelen: Information Sharing on the Semantic Web. Advanced Information and Knowledge Processing, Springer Verlag, 2005.
- [10] D. Fensel: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer Verlag, 2003.

- [11] S. Bechhofer, F. v. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider and L. Stein: OWL web ontology language reference (2004), World Wide Web Consortium.
- [12] M. Klein: Change Management for Distributed Ontologies, PhD thesis, Vrije Universiteit Amsterdam, 2004.
- [13] S. Castano and V.D. Antonellis: A Discovery-Based Approach to Database Ontology Design, *Distrib. Parallel Databases* **7** (1999), 67–98.
- [14] P. Spyns, R. Meersman and M. Jarrar: Data modelling versus ontology engineering, *SIGMOD Rec.* **31** (2002), 12–17.
- [15] A. Gómez-Pérez, M. Fernández-López and O. Corcho: *Ontological Engineering*, Springer Verlag, 2004.
- [16] Y. Ding and S. Foo: Ontology Research and Development Part 1 — A Review of Ontology Generation, *Journal of Information Science* **28** (2002), 123–136.
- [17] R. Tolksdorf and E. Bontas: Engineering a Domain Ontology in a Semantic Web Retrieval System for Pathology, In P. Dadam and M. Reichert, eds.: *GI Jahrestagung* (2), LNI 51 (2004), 569–573.
- [18] M. Erdmann and R. Studer: How to structure and access XML documents with ontologies, *Data Knowl. Eng.* **36** (2001), 317–335.
- [19] E. Wingender: TRANSPATH, TRANSFAC and CYTOMER as starting points for an ontology of regulatory networks, *In Silico Biology* **4**, 2003.
- [20] M. Tresch: *Evolution in Objekt-Datenbanken*, Teubner, Stuttgart, 1995.
- [21] Y. Ding and S. Foo: Ontology research and development, Part 2 - A review of ontology mapping and evolving, *Journal of Information Science* **28** (2002), 375–388.
- [22] M. Ehrig and S. Staab: QOM - Quick Ontology Mapping, In S. McIlraith, D. Plexousakis and F. v. Harmelen, eds.: *Int. Semantic Web Conf. LNCS 3298*, Springer (2004), 683–697.
- [23] C. de Laborda and S. Conrad: Relational.OWL - A Data and Schema Representation Format Based on OWL, In S. Hartmann and M. Stumptner, eds.: *Conceptual Modelling 2005 (APCCM05)*, Australian Computer Society (2005), 89–96.
- [24] R. Bourret, C. Bornhövd and A. Buchmann: A generic load/extract utility for data transfer between xml documents and relational databases, In: *Proc. of the 2nd Int. Workshop on Advance Issues of E-Commerce and Web-Based Information Syst.*, IEEE CS (2000), 134.
- [25] C.B. Necib and J. Freytag: Query Processing Using Ontologies, In O. Pastor and J. Cunha, eds.: *17th Int. Conf. CAiSE 2005. LNCS 3520*, Springer (2005), 167–186.

# A Heuristic Approach to Fragmentation Incorporating Query Information

Hui MA <sup>a,1</sup>, Klaus-Dieter SCHEWE <sup>a</sup> and Markus KIRCHBERG <sup>a</sup>

<sup>a</sup> *Department of Information Systems & Information Science Research Centre  
Massey University, Palmerston North, New Zealand*

**Abstract.** Fragmentation and allocation are database distribution design techniques used to improve the system performance by increasing data localisation and reducing data transportation costs between different network sites. Often fragmentation and allocation are considered separately, disregarding that they are using the same input information to achieve the same objective. Vertical fragmentation is often considered a complicated problem, because the huge number of alternatives make it nearly impossible to obtain an optimal solution. Therefore, many researchers seek for heuristic solutions, among which affinity-based vertical fragmentation approaches form a main stream in the literature. However, using attribute affinities to perform fragmentation can not really reflect the local needs of data at each site. Therefore it is not guaranteed that the remote data transportation costs can be reduced. This paper addresses vertical fragmentation and allocation simultaneously in the context of the relational data model. The core of the paper is a heuristic approach to vertical fragmentation, which uses a cost model and is targeted at globally minimising these costs. Further, based on the proposed vertical fragmentation, an integrated methodology is proposed by applying vertical and horizontal fragmentation simultaneously to produce mixed fragmentation schemata.

**Keywords.** Fragmentation, allocation, relational databases, cost model

## Introduction

In the literature many fragmentation algorithms are affinity based, especially for vertical fragmentation. When vertical fragmentation is discussed in the context of the relational data model (RDM) it takes input of attribute affinities. Hoffer and Severance [1] first cluster attributes according to their pairwise affinity by using the bond energy algorithm (BEA). Navathe *et al* [2] extend the BEA approach and propose a two-phase approach for vertical partitioning. In the first step, they use an Attribute Usage Matrix (AUM) to construct the attribute affinity matrix (AAM) on which clustering is performed. In the second step, estimated cost factors, which reflect the physical environment of fragment storage, are considered to further refine the partitioning schema. Cornell and Yu [3] apply the work of [2] to the physical design of relational databases. Navathe and Ra [4] construct a graph-based algorithm to solve the vertical partitioning problem, where the

---

<sup>1</sup>Corresponding Author: Hui Ma, Department of Information Systems & Information Science Research Centre, Massey University, Private Bag 11 222, Palmerston North, New Zealand; E-mail: h.ma@massey.ac.nz

heuristics used includes an intuitive objective function that is not explicitly quantified. The algorithm presented in Özsu and Valduriez [5] takes AAM as input, employs the BEA to evaluate the togetherness of a pair of attributes, and processes an attribute clustering algorithm. Muthuraj *et al* [6] propose a formal approach to address the problem of an  $n$ -array vertical partitioning problem and derive a partition evaluator function that describes the affinity value for clusters of different sizes. Navathe *et al* [7] propose mixed fragmentation which uses affinities as the main parameters for both horizontal and vertical fragmentation.

To build an attribute affinity matrix, an AUM and an Attribute Access Matrix are used. The latter one shows where the queries are issued. However, during the process of building the AUM site information of the queries is lost. Attribute affinities between a pair of attributes only measure possibilities that attributes are accessed together. Only later, at the stage of allocation, site-specific access measures are employed to determine the allocation of fragments. We argue that the site information should already be used at the stage of fragmentation to meet the needs of data at different sites and to reduce the query processing costs at the same time.

Affinity based fragmentation approaches are also adopted to the object oriented data model (OODM), for which different affinities are used as parameters, e.g., method affinities, attribute affinities, or instance variable affinities, etc. [8,9,10]. However, the deficiencies of the affinity-based algorithms is still there disregarding which model the algorithms are adopted to. In this paper we discuss vertical fragmentation in the context of the relational model to present a cost-based vertical fragmentation heuristic approach, which overcome the deficiencies of the affinity-based algorithms. In a nutshell, we will incorporate all query information, including the site information, by using a simplified cost model at the stage of vertical fragmentation. Doing this way, we can obtain vertical fragmentation and fragment allocation simultaneously with low computational complexity and resulting high system performance. Further we will present integrated methodology which apply vertical and horizontal fragmentation simultaneously to achieve mixed fragmentation schema.

The remainder of this paper is organised as follows. In Section 1 we start with a brief review of fragmentation, allocation and a cost model followed by a discussion of the impact of fragmentation on query costs. In Section 2, we present a heuristic approach to vertical fragmentation based on a simplified cost model, followed by discussions with examples. Horizontal fragmentation for the relational data model are discussed in Section 3, followed with an heuristic approach of mixed fragmentation schema. We conclude with a short summary in Section 4.

## 1. Fragmentation and a Cost Model

Fragmentation and allocation are database distribution design techniques to improve the system performance by reducing the total query costs. In this section we first briefly review fragmentation, including horizontal fragmentation and vertical fragmentation, and then introduce a cost model that can be used to calculate query costs against query trees.

### 1.1. Fragmentation and its Impact on Optimised Query Trees

*Vertical fragmentation* exploits relation schemata to be sets of attributes. It is achieved by performing projection operations on the relation schema. Using Relational Algebra, *vertical fragmentation* could be written as  $R_{Vi} = \pi_{attr(R_{Vi})}(R)$  for all  $i \in \{1, \dots, k_i\}$ . After vertical fragmentation  $R$  is replaced by a set  $\{R_{V1}, \dots, R_{Vk_i}\}$  of new relation schemata, each of which contains a subset of the attributes, i.e.,  $R = \bigcup_{i=1}^{k_i} attr(R_{Vi})$ . Each relation  $r$  over  $R$  is split into relations  $r_{vi} = \pi_{R_{Vi}}(r)$  ( $i = 1, \dots, k_i$ ) such that  $r = r_{v1} \bowtie \dots \bowtie r_{vk_i}$  holds.

In a special case, a distinguishing new attribute *diff* can be added to relation  $R$  as a minimal key to get  $R'$ , then after vertical fragmentation  $diff \in R_{Vi}$  for all  $i \in \{1, \dots, k_i\}$ , and  $R = \pi_{R' - \{diff\}}(R_{V1} \bowtie \dots \bowtie R_{Vk_i})$ . Not having the distinguished new attribute *diff* would require a lossless join-decomposition, which in turn would mean that a join-dependency must hold. However, a well designed database schema would exclude such dependencies, except for the case, where  $R_{V1} \cap \dots \cap R_{Vk_i}$  contains a key. Thus, it is normally required that the primary key (i.e. a chosen minimal key) is part of all  $R_{Vi}$ .

*Horizontal fragmentation* exploit a relation as a set of tuples. It is achieved by performing selection operation on the original relation. By using relational algebra, the operation of horizontal fragmentation can be expressed as:  $R_i = \sigma_{\varphi_i}(R)$  with  $\varphi_i$  as selection predicates defined on  $R$ . After horizontal fragmentation, the relation schema  $R$  are replaced by a set  $\{R_{H1}, \dots, R_{Hk_i}\}$  of new relation schemata, which are the same as  $R$ . The original relation  $r$  over  $R$  can be reconstructed by the union of all the fragments, i.e.,  $r = r_{h1} \cup \dots \cup r_{hk_i}$  with disjoint fragment  $r_{hi} = \sigma_{\varphi_i}(r)$ .

To evaluate a fragmentation schema we analyse the effects of fragmentation on total query costs, for which we focus on query trees. Before fragmentation all leaves of a query tree are relations with their predecessors of leaves as relational algebra. Heuristic query optimisation results in all query trees having sub-queries of the form

$$\pi_X(\sigma_\varphi(R)). \quad (1)$$

For such sub-queries we can always assume that an optimal location assignment will choose the same network nodes for  $\pi_X$ ,  $\sigma_\varphi$  and  $R$ , as the two unary operation will only reduce the size that need to be transferred.

In the query tree vertical fragmentation corresponds to replacing  $R$  with some join  $R_{V1} \bowtie \dots \bowtie R_{Vk_i}$  while horizontal fragmentation corresponds to replacing  $R$  by union  $R = R_{V1} \cup \dots \cup R_{Vk_i}$ . Another round of query optimisation might shift the selection  $\sigma_\varphi$  and projection  $\pi_X$  inside newly introduced join, but the “upper part” of the query tree would not be affected. Therefore, to compare the costs before and after fragmentation it is sufficient and decisive to consider sub-queries in the form of Equation (1) above for the purpose of optimising fragmentation. This discovery is extensively discussed in [11]; it makes the allocation problem simpler than that in [12].

### 1.2. A Cost Model

We now analyse the query costs in the case of fragmentation. The major objective is to base the fragmentation decision on the efficiency of the most frequent queries. As a

general pragmatic guideline we take the recommended rule of thumb to consider only the 20% most frequent queries, as these usually account for most of the data access [5].

Crucial to the query costs are the sizes of relations that have to be built during query execution, as these sets have to be stored at secondary storage, retrieved from there again, and sent between the locations of a network. Therefore, we first approach an estimation of sizes of relations.

Our starting point is a relation schema  $R = \{a_1, \dots, a_n\}$ . Let  $n_r$  denote the average number of tuples in a relation  $r$  over  $R$ , and let  $\ell_i$  denote the average space (in bits) for attribute  $a_i$  in a tuple in  $r$ . The average size of a tuple in  $r$  will be  $\sum_{i=1}^n \ell_i$ , and so the average size of a relation  $r$  will be  $n_r \cdot \sum_{i=1}^n \ell_i$ .

Using these parameters for a vertical fragmentation of  $R$  into  $R_{V1}, \dots, R_{Vk_i}$ , the average size  $s_{vi}$  of a vertical fragment  $R_{Vi}$  will become

$$s_{vi} = \begin{cases} n_r \cdot \sum_{a_i \in R_{Vi}} \ell_i & \text{if } dif \text{ is not used,} \\ n_r \cdot \sum_{a_i \in R_{Vi}} \ell_i + \lceil \log_2 n \rceil & \text{if } dif \text{ is used.} \end{cases}$$

The logarithmic summand in the second cases arises from the fact that we have  $n_r$  different values for the attribute *dif*. If we assume that these are the numbers  $0, 1, \dots, n_r - 1$ , the largest of these numbers will require  $\lceil \log_2 n_r \rceil$  bits in a binary representation.

Using the selection formulae  $\varphi_1, \dots, \varphi_{k_i}$ , horizontal fragmentation of  $R$  split relation  $r$  over  $R$  into fragments  $r_{h1}, \dots, r_{hk_i}$ . Let  $100 \cdot p_i$  be the (average) percentage of tuples in  $r$  satisfying  $\varphi_i$  ( $i = 1, \dots, k_i$ ). Note that we must have  $\sum_{i=1}^{k_i} \varphi_i = 1$ . Then we have  $p_i \cdot n \cdot \sum_{j=1}^k \ell_j$  as the average size of a relation  $r_{hi}$  over  $R_{Hi} = \sigma_{\varphi_i}(R)$  ( $i = 1, \dots, r$ ).

The calculation of sizes of relations applies also to the intermediate results of all queries. However, we can restrict our attention to the nodes in the sub-queries of the form (1), as the other nodes in the query tree will not be affected by horizontal fragmentation and subsequent heuristic query optimisation. Thus, we only have to look at selection and projection nodes and ignore all other nodes in query trees.

- The size of a selection node  $\sigma_\varphi$  is  $p \cdot s$ , where  $s$  is the size of the successor node and  $p$  is the probability that an element in the successor will satisfy  $\varphi$ .
- The size of a projection node  $\pi_X$  is  $(1 - c) \cdot s \cdot \frac{r_i}{r}$ , where  $(1 - c)$  is the probability that any two tuples in  $r$  differs on at least one attribute,  $r$  is the size of relation associated with the successor node, and  $r_i$  is the size of relation associated with the projection node.

Once a fragmentation schema has been decided upon, each fragment must be assigned to one or more nodes in the distributed database management system. The allocation problem involves finding the "optimal" distribution of the fragments to the sites. The discussion of allocation is to find an allocation model that minimises the total costs of processing and storage while trying to meet certain time restrictions [5].

For a given set of fragments  $\{R_1, \dots, R_{k_i}\}$  with different sizes  $s_1, \dots, s_{k_i}$ , if the network has nodes  $N_1, \dots, N_k$ , *fragment allocation* is to assign a node  $N_j$  to each fragment  $R_i$ , which gives rise to a mapping  $\lambda: \{1, \dots, k_i\} \rightarrow \{1, \dots, k\}$ , such that the sum of all the transaction and storage costs from all the sites can be kept to a minimum, where the transaction and storage costs are calculated according to a predefined cost model.

However, the fragments only appear on the leaves of query trees. More generally, we must associate a node  $\lambda(h)$  with each node  $h$  in each relevant query tree.  $\lambda(h)$  indicates the node in the network, at which the intermediate query result corresponding to  $h$  will be stored.

Given a location assignment  $\lambda$  we can compute the total costs of query processing. Let the set of queries be  $Q^m = \{Q_1, \dots, Q_m\}$ . Query costs are composed of two parts: *storage costs* and *transportation costs*:  $cost_\lambda(Q_j) = stor_\lambda(Q_j) + trans_\lambda(Q_j)$ . The storage costs give a measure for retrieving the data back from secondary storage, which is mainly determined by the size of the data. The transportation costs provide a measure for transporting between two nodes of the network.

The storage costs of a query  $Q_j$  depend on the size of the intermediate results and on the assigned locations, which decide the storage cost factors. It can be expressed as

$$stor_\lambda(Q_j) = \sum_h s(h) \cdot d_{\lambda(h)},$$

where  $h$  ranges over the nodes of the query tree for  $Q_j$ ,  $s(h)$  are the sizes of the involved sets, and  $d_i$  indicates the storage cost factor for node  $N_i$  ( $i = 1, \dots, k$ ).

The transportation costs of query  $Q_j$  depend on the sizes of the involved sets and on the assigned locations, which decide the transport cost factor between every pair of sites. It can be expressed by

$$trans_\lambda(Q_j) = \sum_h \sum_{h'} c_{\lambda(h')\lambda(h)} \cdot s(h').$$

Again the sum ranges over the nodes  $h$  of the query tree for  $Q_j$ ,  $h'$  runs over the predecessors of  $h$  in the query tree, and  $c_{ij}$  is a transportation cost factor for data transport from node  $N_i$  to node  $N_j$  ( $i, j \in \{1, \dots, k\}$ ).

Furthermore, for each query  $Q_j$  we assume a value for its frequency  $f_j$ . The total costs of all the queries in  $Q^m$  are the sum of the costs of each query multiplied by its frequency:

$$\sum_{j=1}^m cost_\lambda(Q_j) \cdot f_j.$$

In general, the distribution could be called optimal if we find a fragmentation and allocation schema such that the resulting total query costs are minimal. As this problem is practically incomputable, we suggest to use a heuristic instead.

## 2. A Heuristic Method for Vertical Fragmentation and Allocation

Usually fragmentation and allocation are treated as two isolated problems during the design of distributed databases. During fragmentation no cost model is involved to evaluate the resulting fragmentation schema [2,4,5]. However, once the fragmentation decision

has been made, the possibility of finding the optimal allocation schema of the fragments is restricted. Therefore, a cost model should come into play when we make decision of fragmentation. In this section we concentrate on vertical fragmentation.

As shown in the Example 1 in [13], affinity-based vertical fragmentation can not really get an optimal solution for fragmentation because it does not reflect the local needs of data of a relation. The chance of finding an optimal fragmentation and allocation is restricted if fragmentation is performed as an separate procedure without considering how it affect total query costs. We observe that the fragmentation and allocation in Scenario IV is the best solution, which results from comparing with query costs while making decision of fragmentation. That is, with the target of improving system performance, to get an optimal solution of fragmentation and allocation we need to employ a cost model, with which we can achieve an optimal fragmentation and allocation simultaneously. With the cost model, any change of the query information, including the site information of queries, will be reflected in the design of fragmentation and allocation. But the approaches of vertical fragmentation using the value of affinities can not reflect this change.

However, if a relation has  $m$  non-primary key attributes, the possible fragments are given by the *Bell* number which is approximately  $B(m) \approx m^m$ . With this number of possible fragments, the following up allocation, using the cost model introduced previously, is of the complexity  $k^{(m^m)}$  with  $k$  as the number of network nodes. Therefore, it is impossible to get the optimal solutions to the vertical fragmentation and allocation problems. We can only expect to find a heuristic solution. In the remaining of this section, we first define some terms. Then we introduce a heuristic approach to vertical fragmentation and allocation. This approach adapts a simplified cost model while making decision of vertical fragmentation.

### 2.1. Requests and Needs at Sites

We now define some terms that will be used in our discussion. Assume a relation  $R = \{a_1, \dots, a_n\}$  being accessed by a set of queries  $Q^m = \{Q_1, \dots, Q_j, \dots, Q_m\}$  with frequencies  $f_1, \dots, f_m$ , respectively. To improve the system performance, relation  $R$  is vertically fragmented into a set of fragments  $\{R_{V1}, \dots, R_{Vu}, \dots, R_{Vki}\}$ , each of which is allocated to one of the network nodes  $N_1, \dots, N_\theta, \dots, N_k$ . Each attribute  $a_i$  of  $R$  is of average length  $\ell_i$ . Note that the maximum number of fragments is  $k$ , i.e.,  $k_i \leq k$ . We use  $\lambda(Q_j)$  to indicate the site that issues query  $Q_j$ , and use  $A_j = \{a_i | f_{ji} = f_j\}$  to indicate the set of attributes that are accessed by  $Q_j$ , with  $f_{ji}$  as the frequency of the query  $Q_j$  accessing attributes  $a_i$ . Here,  $f_{ji} = f_j$  if the attribute  $a_i$  is accessed by  $Q_j$ . Otherwise,  $f_{ji} = 0$ .

From the discussion in the previous section we know that to get an optimal vertical fragmentation we need to employ a cost model which takes input information as:

- the frequency of queries that access the object; when the same query is issued at different sites, it is treated as different queries;
- the subset of the attributes used by queries;
- the size of each attribute of the object;
- the site that issue the queries.

To record the above input information we introduce *Attribute Usage Frequency Matrix* (AUFM). Each row represents one query  $Q_j$ ; the head of column is the set of at-



tributes of a relation schema  $R$ . In addition, there are two columns with one column indicating the site that issues the queries and the other indicating the frequency of the queries. The values on a column indicate the frequency  $f_{ji}$  of the query  $Q_j$  that use the corresponding attributes  $a_i$  grouped by the site that issues queries. Note that we treat the same query at different sites as different queries. Doing this way we only need one matrix to record all the information rather than two matrices, Attribute Usage Matrix and Access Matrix, that are used in [4]. Subsequently, the following up calculation is easy to be formulated.

From one site each attribute is requested by multiple queries. The *request* of an attribute at a site  $\theta$  is the sum of frequencies of all queries at the site accessing the attribute. It can be calculated with the formula below:

$$request_{\theta}(a_i) = \sum_{j=1, \lambda(Q_j)=\theta}^m f_{ji}.$$

With the length  $\ell_i$  of an attribute  $a_i$ , we can calculate the *need* of an attribute as the total data volume involved to retrieve  $a_i$  by all the queries:

$$need_{\theta}(a_i) = \ell_i \cdot \sum_{j=1, \lambda(Q_j)=\theta}^m f_{ji} = \ell_i \cdot request_{\theta}(a_i).$$

The *need* of a fragment  $F_u$  at a site  $\theta$  is calculated with the following formula:

$$\begin{aligned} need_{\theta}(F_u) &= \sum_{j=1, \lambda(Q_j)=\theta}^m s_j \cdot f_j \\ &= \sum_{j=1, \lambda(Q_j)=\theta}^m \left( \sum_{i=1}^n \ell_i \cdot f_{ji} \right) = \sum_{i=1}^n need_{\theta}(a_i). \end{aligned}$$

with  $s_j$  as the size of data volume required by query  $Q_j$  from fragment  $F_u$ .

Finally, we introduce a term *pay* to measure the costs of accessing a single attribute once it is allocated to a network node. The *pay* of allocating an attribute  $a_i$  to a site  $\theta$  measures the costs of accessing attribute  $a_i$  by all queries from the other sites  $\theta'$ , which is different from  $\theta$ . It can be calculated using the following formula:

$$pay_{\theta}(a_i) = \sum_{\theta'=1, \theta \neq \theta'}^k request_{\theta'}(a_i) \cdot c_{hh'}.$$

We do not include attribute length in the formula because when we compare the *pay* of an attribute at different sites, attribute length will always be the same.

As in [2], we assume a simple transaction model, using which the system collects the information at the site of the query and executes the query there. In this case we can evaluate costs of allocating a single attribute to network nodes and then make decision by choosing a site that leads to the least query costs. Also, according to our discussion of how fragmentation affect query costs, the allocation of fragments to network nodes, following the cost minimisation heuristics, already determine the location assignment provided that an optimal location assignment for the queries was given prior to the fragmentation.

In distributed databases, costs of queries are dominated by the cost of data transportation from a remote site to the site that issued the queries. To compare different vertical fragmentation schemata we would like to compare how it affect the transportation costs. So we can simplify the cost model in Section 1.2 as following:

$$cost_{\lambda}(Q^m) = \sum_{\theta=1}^k \sum_{u=1}^v need_{\theta}(F_u) \cdot c_{\theta\theta'}. \quad (2)$$

Note that the cost factor  $c_{\theta\theta'} = 0$ , if  $\theta = \theta'$ .

## 2.2. Heuristics for Vertical Fragmentation

Taking the simplified cost model introduced above we now analyse the relationships between *cost*, the *pay* and the *request* of an attribute. We compute the following fomulae:

$$\begin{aligned} cost_{\lambda(a_i)=\theta}(Q^m) &= \sum_{\theta'=1, \theta' \neq \theta}^k need_{\theta'}(a_i) \cdot c_{\theta\theta'} \\ &= \sum_{\theta'=1, \theta' \neq \theta}^k \sum_{j=1, \lambda(Q_j)=\theta'}^m \ell_i \cdot f_{ji} \cdot c_{\theta\theta'} \\ &= \ell_i \cdot \sum_{\theta'=1, \theta' \neq \theta}^k \sum_{j=1, \lambda(Q_j)=\theta'}^m f_{ji} \cdot c_{\theta\theta'} \\ &= \ell_i \cdot \sum_{\theta'=1, \theta' \neq \theta}^k request_{\theta'}(a_i) \cdot c_{\theta\theta'} \\ &= \ell_i \cdot pay_{\theta}(a_i). \end{aligned}$$

The above formula gives rise to two alternative heuristics for the allocation of an attribute  $a_i$  ( $i = 1, \dots, n$ ).

- The first heuristic allocates  $a_i$  to a network node  $N_w$  such that  $pay_w(a_i)$  is minimal, i.e., we choose a network node in such a way that the total transport costs for all queries arising from the allocation are minimised.
- The second heuristic allocates  $a_i$  to a network node  $N_w$  such that  $request_w(a_i)$  is maximal, i.e., we choose the network node with the highest *request* of the attribute  $a_i$ . This guarantees that there are no transport costs associated with data of attribute  $a_i$  for those queries that need the data of  $a_i$  most frequently. In addition, the availability of data of attribute  $a_i$  will be maximised.

Taking the first heuristic we perform vertical fragmentation with the following steps. We do not distinguish read and write queries because replication is not considered at this stage. The second heuristic is easy to be formulated.

1. Take the most frequently used 20% queries  $Q^N$ .
2. Optimise all the queries and construct an *AUFM* for each database type  $R$  based on the queries.

3. Calculate the *request* at each site for each attribute to construct an Attribute *request* Matrix.
4. Calculate the *pay* at each site for each attribute to construct an Attribute *pay* Matrix.
5. Cluster all attributes to the site which has the lowest value of the *pay*.
6. Attach the primary key to each of the fragments.

This procedure is formally described as the algorithm below.

**Algorithm 1 (Vertical Fragmentation)**

**Input:**  $Q^N = \{Q_1, \dots, Q_n\}$  /\* a set of queries

$R = \{a_1, \dots, a_n\}$  /\* a type with a set of attributes

a set of network nodes  $N = \{1, \dots, k\}$

The *AUFM* of  $R$

**Output:** vertical fragmentation schema and fragment allocation schema,  $\{R_{V1}, \dots, R_{Vk}\}$

**Begin**

```

for each  $\theta \in \{1, \dots, k\}$  let  $attr(R_{V\theta}) = id(R)$  endfor
for each attribute  $a_i \in R, 1 \leq i \leq n$  do
  for each node  $\theta \in \{1, \dots, k\}$ 
    do calculate  $request_{\theta}(a_i)$ 
  endfor
  for each node  $\theta \in \{1, \dots, k\}$ 
    do calculate  $pay_{\theta}(a_i)$ 
  endfor
  choose  $w$  such that  $pay_w(a_i) = \min_{\theta=1}^k pay_{\theta}(a_i)$  /* find the minimum
   $attr(R_{Vw}) = attr(R_{Vw}) \cup a_i$ , /* add  $a_i$  to  $R_{Vw}$ 
endfor
for each  $\theta \in \{1, \dots, k\}, R_{V\theta} = \pi_{attr(R_{V\theta})}(R)$  endfor

```

The above algorithm first finds the site that has the smallest value of the *pay* then allocates the attribute to the site. A vertical fragmentation and allocation schema are obtained simultaneously.

Due to the space limitation we take the simple motivation example in [13] to show how the above algorithm is applied to perform vertical fragmentation. The query information given are two queries,  $Q_1$  and  $Q_2$ , accessing fragment  $F_u = \{a_1, a_2, a_3, a_4\}$ , with frequencies  $f_1, f_2$  respectively. For Scenario IV, where  $Q_1$  and  $Q_2$  are issued at site  $N_1$  and  $N_2$ , respectively, we construct attribute usage frequency matrix as Table 1. We then construct attribute *request* matrix as Table 2. Using the values of the *request* in Table 2 and the values for transportation cost factors,  $c_{12}$  and  $c_{21}$ , we get the attribute *pay* matrix as in Table 3.

As  $c_{12} = c_{21}$ , to allocate attribute to the site of the lowest value of *pay* we need to compare the values of the *request* at the two sites. We get  $\lambda(a_1) = 1, \lambda(a_2) = 1, \lambda(a_3) = 2, \lambda(a_4) = 2$ . Therefore, grouping attributes according the sites we get two

**Table 1.** Attribute usage frequency matrix

Site	Query	Frequency	$a_1$	$a_2$	$a_3$	$a_4$
1	$Q_1$	$f_1$	$f_1$	$f_1$	$f_1$	$f_1$
2	$Q_2$	$f_2$	0	0	$f_2$	$f_2$

**Table 2.** Attribute request matrix

Attribute	$a_1$	$a_2$	$a_3$	$a_4$
$request_1$	$f_1$	$f_1$	$f_1$	$f_1$
$request_2$	0	0	$f_2$	$f_2$

**Table 3.** Attribute pay matrix

Attribute	$a_1$	$a_2$	$a_3$	$a_4$
$pay_1$	0	0	$f_2 \cdot c_{21}$	$f_2 \cdot c_{21}$
$pay_2$	$f_1 \cdot c_{12}$	$f_1 \cdot c_{12}$	$f_1 \cdot c_{12}$	$f_1 \cdot c_{12}$

fragments  $F_{u1} = \{a_1, a_2\}$ ,  $F_{u2} = \{a_3, a_4\}$ , which are allocated to network nodes  $N_1$  and  $N_2$ , respectively. This is the same as Scenario IV in [13], the optimised fragmentation and allocation that are achieved by using the cost model.

A more comprehensive example is presented in [13] to compare our approach with some existing well-known affinity based approaches by applying our proposed vertical fragmentation algorithm to the same problem example in [2,4]. The results show that using the same set of queries as input we can achieve the same vertical fragmentation schema within shorter time, which means our approach is efficient in terms of complexity. Further, we observe that the changes of the issuing site of one query to a different site leads to the resulted vertical fragmentation schema changed using our approach, which is reasonable because the change of input query information indicate the change of data needs at different sites. However, the existing affinity-based approach does not lead to different vertical fragmentation schema because the changes of issuing site does not change any values of affinities. With the cost model we show that our approach leads to less query costs.

### 2.3. Discussion

The advantages of our heuristic approach for vertical fragmentation and fragment allocation are:

- Except key attributes, there is no overlap among all the vertical fragments. Therefore, we do not need extra procedure to remove overlaps.
- The change of queries will be reflected by the fragmentation solution. Query information may reflect the needs to retain attributes from some sites more often than some other sites. Even though on the affinity graph the cutting edges will be the same.
- The complexity of this approach is low. Lets  $m$  be the number of queries,  $n$  be the number of attributes,  $k$  be the number of network nodes. The complexity of our approach, which dealing with vertical fragmentation and allocation, is  $O(m \cdot n + k^2 \cdot n)$ , while the complexity of graphical approach in [4] is  $O(n^2 \cdot m + k^n)$  for the whole design procedure, including building the affinity matrix, vertical fragmentation and allocation.
- This approach suits the situation that for each relation the number of attributes is small and the number of queries is big. Usually, the number of queries taken into consideration is bigger than the number of attributes of a relation.

### 3. An Integrated Approach of Horizontal Fragmentation and Vertical Fragmentation

Database distribution design often involves both horizontal and vertical fragmentation to meet the requirement of queries, which access a subset of attributes and a subset of tuples of a relation. This requires us to perform mixed fragmentation, where each fragment belongs to one horizontal and one vertical fragmentation. In this section we propose an integrated design methodology which takes into consideration of query information while making decision of mixed fragmentation. The design methodology presented in this section integrates the horizontal fragmentation algorithm in [14] with the vertical algorithm presented in the above section.

#### 3.1. An Heuristic Approach to Horizontal Fragmentation for the Relational Databases

For the above purpose we first adopt the heuristic approach of horizontal fragmentation algorithm from [14] to the RDM. Let  $\Phi^m = \{\varphi_1, \dots, \varphi_m\}$  denote a set of simple predicates defined on relation schema  $R$ . Then the set of *normal predicates*  $\mathcal{N}^m = \{\mathcal{N}_1, \dots, \mathcal{N}_n\}$  on relation schema  $R$  is the set of all satisfiable predicates of the form  $\mathcal{N}_j \equiv \varphi_1^* \wedge \dots \wedge \varphi_m^*$ , where  $\varphi_i^*$  is either  $\varphi_i$  or  $\neg\varphi_i$ .

Each of the normal predicates defines an *atomic horizontal fragment*,

$$F_j = \sigma_{\mathcal{N}_j}(R).$$

Let  $J \subseteq \{1, \dots, m\}$  be a set of indices of a subset of all simple predicates. Normal predicates can be represented with the following form :

$$\mathcal{N}_j = \bigwedge_{j \in J} \varphi_i \wedge \bigwedge_{i \notin J} \neg\varphi_i.$$

Let  $f_j$  be the frequency of predicate  $\varphi_j$ ,  $J_\theta = \{j | j \in J \wedge \varphi_j \text{ executed at site } \theta\}$  be a subset of indices of all simple predicates, executed at site  $N_\theta$ . We now define *request* of an atomic fragment at site  $\theta$  as the sum of frequencies of predicates issued at site  $\theta$ :

$$request_\theta(F_j) = \sum_{j=1, j \in J_\theta}^k f_j.$$

We introduce term *pay* of allocating an atomic horizontal fragments at a site  $\theta$  as the costs of accessing the atomic horizontal fragment by all queries from sites other than  $\theta$ :

$$pay_\theta(F_j) = \sum_{\theta'=1, \theta' \neq \theta}^k request_{\theta'}(F_j) \cdot c_{\theta\theta'}.$$

We now present the procedure of horizontal fragmentation as the following:

1. sort queries by decreasing frequency to get a list of queries  $\mathcal{Q} = [Q_1, \dots, Q_j, \dots, Q_m]$
2. optimise all the queries and extract simple predicates from the queries to get a list of  $\Phi$  of simple predicates,
3. construct a usage matrix based on  $\Phi$  to obtain a list of simple predicates  $\Phi^w$  for each relation schema  $R$ ,

4. take the list of selection predicates  $\Phi^w = [\varphi_1, \dots, \varphi_w]$  to get a list of indices  $X = [0, 1, \dots, x_1, \dots, x_2, \dots, w]$ ,
5. using *Num\_Simple\_Predicates* in [14] to get value  $y$ , the number of simple predicates to be used for horizontal fragmentation,
6. take the first  $y$  simple predicates in  $\Phi^w$  to get a subset of simple predicates  $\Phi^y$ ,
7. perform horizontal fragmentation using Algorithm 2 as shown below.

### Algorithm 2 (Horizontal Fragmentation)

**Input:**  $\Phi^y = \{\varphi_1, \dots, \varphi_y\}$  /\* a set of simple predicates

**Output:** Horizontal fragmentation schema and fragment allocation schema

**Begin**

```

for each  $\theta \in \{1, \dots, k\}$ 
   $R_{H\theta} = \emptyset$ 
endfor
define a set of normal predicates  $\mathcal{N}^y$  using  $\Phi^y$ 
define a set of atomic horizontal fragments  $\mathcal{F}^y$  using  $\mathcal{N}^y$ 
for each atomic fragment  $F_j \in \mathcal{F}^y, 1 \leq i \leq 2^y$  do
  for each node  $\theta \in \{1, \dots, k\}$  do
    calculate  $request_\theta(F_j)$ 
    calculate  $pay_\theta(F_j)$ 
  endfor
  choose  $w$  such that  $pay_w(F_j) = \min(pay_1(F_j), \dots, pay_k(F_j))$ 
  /* find the minimum value
   $\lambda(F_j) = \mathcal{N}_w$  /* allocate the atomic fragment to the site of the smallest  $pay$ 
  define  $R_{H\theta}$  with  $R_{H\theta} = \bigcup \{F_j : \lambda(F_j) = N_\theta\}$ 
  /* put the atomic fragment into the corresponding fragment
endfor

```

The above algorithm first finds the site that has the biggest value of  $pay$  then allocates the atomic fragment to the site. A fragmentation schema and fragment allocation schema can be obtained simultaneously.

### 3.2. An Heuristic Approach to Mixed Fragmentation

Mixed fragmentation is a process of simultaneously applying horizontal and vertical fragmentation on a relation [7]. Affinity-based mixed fragmentation approach is proposed in [7], with horizontal fragmentation taking predicate affinities as input while vertical fragmentation taking attribute affinities as input. Due to the deficiencies of affinity-based fragmentation approaches pointed out in [13], we now propose a heuristic approach to mixed fragmentation, which performs horizontal fragmentation and vertical fragmentation simultaneously based on a cost model.

Ideally to achieve an optimal fragmentation and allocation, we could perform mixed fragmentation using a grid with each tuple of the grid as an atomic horizontal fragment and columns as the set of attributes of the relation. The grid cells are then grouped at the site of the lowest value of  $pay$  to form mixed fragments so as to increase the data local availability and to reduce remote transportation costs. However, this will lead to a set of irregular fragments as in [7], which will be difficult to be presented as a single relation

without introducing null values. We have to seek for a design methodology to get a set of regular fragments. The following procedure integrates our proposed algorithms for horizontal and vertical fragmentation to get a set of mixed fragmentation schema.

1. Simultaneously apply both horizontal and vertical fragmentation on a relation  $R$  to get a grid. Performing horizontal fragmentation, using algorithm 2, results a set of horizontal fragments,  $\{R_{H1}, \dots, R_{Hk}\}$ , while performing vertical fragmentation, using algorithm 1 introduced above, results a set of vertical fragments  $\{R_{V1}, \dots, R_{Vk}\}$ . The set of grid cells are represented as  $R_{H1V1}, R_{H1V2}, \dots, R_{H\alpha V\beta}, \dots, R_{kk}$ , each of which belongs to exactly one horizontal and one vertical fragment of the relation.
2. Group the grid cells from the above step to the site of the smallest *request*. The first two indices of each grid cell indicate the horizontal fragment it belongs to and also the site allocation that leads to the lowest query costs. Analogously, the last two indices indicate the vertical fragment it belongs to and the site of the lowest costs. For a grid cell which belongs to horizontal and vertical fragments of the same allocation, e.g.,  $R_{H\theta V\theta}$ , we allocate it to the site  $\theta$ . For the grid cells with different indices for horizontal and vertical fragments, we choose the allocation of the biggest *request*. For example, for a cell  $R_{H\alpha V\beta}$ , if  $request_{\alpha}(R_{H\alpha V\beta}) > request_{\beta}(R_{H\alpha V\beta})$  allocate  $R_{H\alpha V\beta}$  to site  $\alpha$ .
3. Merge fragment cells that are of the same horizontal or vertical fragment if they are allocated to the same site. Fragment cells of the same vertical fragment are merged though join operation, i.e.,  $R_{H\alpha V\beta} \bowtie R_{H\alpha V\alpha}$  if  $R_{H\alpha V\beta}$  is allocated to site  $\alpha$ . Fragment cells of the same horizontal fragment are merged through union, i.e.,  $R_{H\alpha V\beta} \cup R_{H\beta V\beta}$  if they are allocated to site  $\beta$ .

*Remark:* It is reasonable to decide the allocation of a grid cell  $R_{H\alpha V\beta}$  according to the values of *request* at the site  $\alpha$  and  $\beta$ . At the step of horizontal fragmentation, fragments are allocated to site  $\alpha$  because  $pay_{\alpha}$  is the smallest among all sites. This means the total query costs of accessing the fragment through predicates are smallest if it is allocate to site  $\alpha$ . At the step of vertical fragmentation, an attributes is allocated to site  $\alpha$  because it has the lowest  $pay_{\alpha}$ . In other words the total query costs of accessing it through projection is the lowest if it is at site. Therefore the optimal allocation of  $R_{H\alpha V\beta}$  should be either  $\alpha$  or  $\beta$ . To compare the total query costs of allocating the fragment cell at  $\alpha$  and  $\beta$  we can compare the total query costs which involves transportation cost factors between the two sites,  $\alpha$  and  $\beta$ . Because  $c_{\alpha} = c_{\beta}$ , using the simplified cost model as in 2.1 we only need to compare values of the *request* at the two sites.

#### 4. Conclusion

In this paper we presented a heuristic approach to vertical fragmentation for the relational data model. The major objective is to provide a tractable approach to minimising the query processing costs for the most frequent queries. In general, this would require to consider all possible fragmentations and all possible allocations of intermediate query results to the nodes of a network, which is intractable. Instead of this we suggest to consider vertical fragmentation and allocation using a cost model at the same time. In addition, we integrate the handing of horizontal fragmentation, which has been discussed

in [14], and vertical fragmentation with the consideration of the requirement of global optimisation.

The next step of our work is to adapt the heuristic approach to complex value databases, which cover the common aspects of object-oriented databases, object-relational databases, and databases based on the eXtensible Markup Language (XML). For this, complex data types should be covered by the cost model. The procedure of vertical fragmentation should be altered accordingly.

## References

- [1] Hoffer, J.A., Severance, D.G.: The use of cluster analysis in physical database design. In: *Proceedings of the First International Conference on Very Large Data Bases*. (1975), 69–86.
- [2] Navathe, S.B., Ceri, S., Wiederhold, G., Dour, J.: Vertical partitioning algorithms for database design. *ACM TODS* **9**(4) (1984), 680–710.
- [3] Cornell, D., Yu, P.: A vertical partitioning algorithm for relational databases. In: *International Conference on Data Engineering*, Los Angeles, California (1987), 30–35.
- [4] Navathe, S.B., Ra, M.: Vertical partitioning for database design: A graphical algorithm. *SIGMOD Record* **14**(4) (1989), 440–450.
- [5] Özsu, M.T., Valduriez, P.: *Principles of Distributed Database Systems*. Alan Apt, New Jersey (1999).
- [6] Muthuraj, J., Chakravarthy, S., Varadarajan, R., Navathe, S.B.: A formal approach to the vertical partitioning problem in distributed database design. In: *Proceedings of the Second International Conference on Parallel and Distributed Information Systems*, San Diego, CA, USA (1993), 26–34.
- [7] Navathe, S., Karlapalem, K., Ra, M.: A mixed fragmentation methodology for initial distributed database design. *Journal of Computer and Software Engineering* **3**(4) (1995).
- [8] Bellatreche, L., Simonet, A., Simonet, M.: Vertical fragmentation in distributed object database systems with complex attributes and methods. In R. Wagner, H.T., ed.: *Seventh International Workshop on Database and Expert Systems Applications, DEXA '96*, IEEE-CS Press (1996), 15–21.
- [9] Ezeife, C.I., Barker, K.: Vertical fragmentation for advanced object models in a distributed object based system. In: *Proceedings of the 7th International Conference on Computing and Information*. (1995), 613–632.
- [10] Karlapalem, K., Navathe, S.B., Morsi, M.M.A.: Issues in distribution design of object-oriented databases. In: *IWDOM*. (1992), 148–164.
- [11] Ma, H., Schewe, K.D., Wang, Q.: Distribution design for higher-order data models. *Data and Knowledge Engineering* (2007), to appear.
- [12] Apers, P.M.G.: Data allocation in distributed database systems. *ACM Transactions on Database Systems* **13** (1988), 263–304.
- [13] Ma, H., Schewe, K.D., Kirchberg, M.: A heuristic approach to vertical fragmentation incorporating query information. In Vasilecas, O., Eder, J., Caplinskas, A., eds.: *Proceedings of the 7th International Baltic Conference on Databases and Information Systems*, IEEE (2006), 69–76.
- [14] Ma, H., Schewe, K.D., Wang, Q.: A heuristic approach to cost-efficient fragmentation and allocation of complex value databases. In J. Bailey, G.D., ed.: *Proceedings of the 17th Australian Database Conference*. CRPIT 49 (2006), 119–128.



# Introducing Softness into Inductive Queries on String Databases

Ieva MITASIUNAITE and Jean-François BOULICAUT  
INSA Lyon, LIRIS CNRS UMR 5205, 69621 Villeurbanne cedex, France  
{Ieva.Mitasiunaite,Jean-Francois.Boulicaut}@insa-lyon.fr

**Abstract.** In many application domains (e.g., WWW mining, molecular biology), large string datasets are available and yet under-exploited. The inductive database framework assumes that both such datasets and the various patterns holding within them might be queryable. In this setting, queries which return patterns are called inductive queries and solving them is one of the core research topics for data mining. Indeed, constraint-based mining techniques on string datasets have been studied extensively. Efficient algorithms enable to compute complete collections of patterns (e.g., substrings) which satisfy conjunctions of monotonic and/or anti-monotonic constraints in large datasets (e.g., conjunctions of minimal and maximal support constraints). We consider that fault-tolerance and softness are extremely important issues for tackling real-life data analysis. We address some of the open problems when evaluating soft-support constraint which implies the computations of pattern soft-occurrences instead of the classical exact matching ones. Solving efficiently soft-support constraints is challenging since it prevents from the clever use of monotonicity properties. We describe our proposal and we provide an experimental validation on real-life clickstream data which confirms the added value of this approach.

**Keywords.** Inductive databases, fault-tolerance, sequence mining

## Introduction

Collecting huge volumes of sequential data (i.e., the data is a collection of sequences or strings in a given alphabet) has become far easier in many application domains (e.g., E-commerce, networking, life sciences). Our ability to discover actionable patterns from such datasets remains however limited.

This paper focuses on substring mining, i.e., the searched patterns are strings as well. Knowledge discovery processes based on substrings in string datasets have been studied extensively. We study a database perspective on such processes, the so-called inductive database approach [1,2,3,4]. The idea is that many steps in complex knowledge discovery processes might be considered as queries which returns selected data instances and/or patterns holding in the data. Designing query languages which would support such a querying activity remains a long-term goal, and only preliminary results have been obtained for some quite specific scenarios (e.g., rather simple processes based on association rule mining [5]). As a needed step towards query languages for inductive databases,

it is interesting to investigate different pattern domains and this chapter is considering inductive queries on the string pattern domain. Such queries declaratively express the constraints that have to be satisfied by the solution patterns. Typical challenges are (a) to identify useful primitive constraints to specify the a priori interestingness of the patterns in the data, and (b) to be able to design efficient and (when possible) complete solvers for computing every pattern which satisfies a combination of primitive constraints.

The state-of-the-art is that efficient algorithms are available for solving specific conjunctions of primitive constraints on string patterns. For instance, many solvers have been designed for frequent substring or sequential patterns possibly combined with some more or less restricted types of syntactic constraints (e.g., [6,7,8,9,10,11]). A promising approach has been developed by De Raedt and colleagues which consider arbitrary Boolean combination of primitive constraints which are either monotonic or anti-monotonic [12,13,14]. Indeed, a key issue for designing efficient solvers is to consider constraint properties (like anti-monotonicity and its dual monotonicity property) and exploit them for clever search space pruning. Many useful primitive constraints are monotonic (e.g., maximal support in a data set, enforcing a given sub-string occurrence) or anti-monotonic (e.g., minimal support, avoiding a given sub-string occurrence).

Some useful constraints are however neither anti-monotonic nor monotonic. This is the case of regular expression constraints, i.e., rich syntactic constraints which enforce the solution patterns to belong to the language of a given regular expression. A typical application in WWW usage mining would be to look for the frequent sequences of clicks which are matching a given path through the WWW site (conjunction of a minimal support constraint with a regular expression constraint). Efficient ad-hoc optimization strategies have been developed for such a conjunction [15,16,17]. In many applications, it is also interesting to look for patterns which are similar enough to a reference pattern. For instance, it might be useful to look for sequences of clicks on a WWW site which are frequent for a given group of users, infrequent for another group and which are similar enough to an expected pattern specified by the WWW site designer. Such a primitive similarity constraint generally lacks from any monotonicity property. In [18], we have studied a possible decomposition of such a constraint into a conjunction of an anti-monotonic one and a monotonic one. It was implemented on top of the FAVST algorithm [14] such that it is now possible to combine efficiently that kind of similarity constraint with other monotonic or anti-monotonic user-defined constraints.

This paper follows this direction of research and considers the intrinsic limitations of these previous approaches which are all based on exact matching of candidate patterns with data instances. We are indeed interested in string database analysis for various application domains (e.g., WWW usage mining, seismic or telecommunication data analysis, molecular biology). Even though our raw data is fundamentally sequential (spatially or temporally ordered), the strings to be mined are generally preprocessed: the data can be fundamentally noisy due to technological issues w.r.t. measurement, alphabet design can be somehow exploratory, but also the phenomena we would like to observe can be fundamentally fuzzy and such that soft computing approaches are needed.

In many application domains, the string alphabet has to be designed and/or computed. For instance, in a WWW usage mining context, assume that the raw data concern facts about “Users who performed Operations from Machines”. Depending of the analysis task, many event types and thus alphabets might be considered (e.g., an operation is performed from a machine, a user has performed something, a user has per-

formed something from a machine) and a meaningful browsing sequence will often be some kind of consensus between different occurrences of similar browsing sequences. Also, in many cases of sequential data mining, data are available as numerical time series that can be analyzed by means of substring pattern algorithms provided that the data is discretized and thus encoded as a sequence of events in a “computed” alphabet. These methods are not always robust enough and again, soft-occurrences of patterns might appear much more meaningful. Finally, the most famous case of degenerated data concerns molecular biology. We can consider that patterns on gene promoter sequences (e.g., substrings within DNA sequences) play a major role in gene regulation but it is well-known that evolution has lead to many variants of the “originally” useful patterns. As a result, when looking at the major scientific question of transcription factor binding site in DNA sequences, molecular biologists consider consensus regular expressions instead of exact matching information over the gene promoter sequences.

In this paper, we address some of the open problems when computing soft-occurrences of patterns within string dataset. This is a significant revision and extension of the paper [19]. For instance, an original and detailed experimental validation has been carried out to replace the preliminary results described in [19]. In Section 1, we provide the needed definitions and the problem setting. Section 2 introduces our definition of soft-occurrences and our formalization of soft-support constraints. The proofs of properties are available and can be asked to the authors. They are omitted here because of space limitation. In Section 4, we provide an in-depth experimental validation on real-life clickstream data which confirms the added value of our approach. Finally, Section 5 is a short conclusion.

## 1. Problem Setting

**Definition 1 (Basic notions on strings)** Let  $\Sigma$  be a finite alphabet, a string  $\sigma$  over  $\Sigma$  is a finite sequence of symbols from  $\Sigma$ , and  $\Sigma^*$  denotes the set of all strings over  $\Sigma$ .  $\Sigma^*$  is our language of patterns  $\mathcal{L}$  and we consider that the mined data set denoted  $r$  is a multi-set<sup>1</sup> of strings built on  $\Sigma$ .  $|\sigma|$  denotes the length of a string  $\sigma$  and  $\epsilon$  denotes the empty string. We note  $\sigma_i$  the  $i^{\text{th}}$  symbol of a string  $\sigma$ ,  $1 \leq i \leq |\sigma|$ , so that  $\sigma = \sigma_1\sigma_2 \dots \sigma_{|\sigma|}$ . A sub-string  $\sigma'$  of  $\sigma$  is a sequence of contiguous symbols in  $\sigma$ , and we note  $\sigma' \sqsubseteq \sigma$ .  $\sigma$  is thus a super-string of  $\sigma'$ , and we note  $\sigma \supseteq \sigma'$ . We assume that, given a pattern  $\phi \in \mathcal{L}$ , the supporting set of strings in  $r$  is denoted by  $\text{ext}(\phi, r) = \{\sigma \in r \mid \phi \sqsubseteq \sigma\}$ .

**Example 1** Let  $\Sigma = \{a, b, c, d\}$ .  $abbc$ ,  $abdbc$ ,  $\epsilon$  are examples of strings over  $\Sigma$ . Examples of sub-strings for  $abdbc$  are  $a$  and  $dbc$ .  $aabdbcd$  is an example of a super-string of  $abdbc$ . If  $r$  is  $\{abccb, adccba, ccabd\}$ ,  $\text{ext}(ccb, r) = \{abccb, adccba\}$ .

**Definition 2 (Inductive queries)** A constraint is a predicate that defines a property of a pattern and evaluates either to true or false. An inductive query on  $\mathcal{L}$  and  $r$  with parameters  $p$  is fully specified by a constraint  $Q$  and its evaluation needs the computation of  $\{\phi \in \mathcal{L} \mid Q(\phi, r, p) \text{ is true}\}$  [20]. In the general case,  $Q$  is a Boolean combination of the so-called primitive constraints.

<sup>1</sup>Data may contain multiple occurrences of the same sequence.

**Definition 3 (Generalisation/specialisation)** A pattern  $\phi$  is more general than a pattern  $\psi$  (denoted  $\phi \succeq \psi$ ) iff  $\forall r \text{ ext}(\phi, r) \supseteq \text{ext}(\psi, r)$ . We also say that  $\psi$  is more specific than  $\phi$  (denoted  $\psi \preceq \phi$ ). Two primitive constraints can be defined: *MoreGeneral*( $\phi, \psi$ ) is true iff  $\phi \succeq \psi$  and *MoreSpecific*( $\phi, \psi$ ) is true iff  $\phi \preceq \psi$ .

For strings, constraint *SubString*( $\phi, \psi$ )  $\equiv \phi \sqsubseteq \psi$  (resp., *SuperString*( $\phi, \psi$ )  $\equiv \phi \supseteq \psi$ ) are instances of *MoreGeneral*( $\phi, \psi$ ) (resp., *MoreSpecific*( $\phi, \psi$ )). In other terms,  $\forall \phi, \psi \in \mathcal{L}$ ,  $\phi \succeq \psi$  iff  $\phi \sqsubseteq \psi$ . Given a threshold value  $n$ , one can limit a maximal number of occurrences of  $\psi$  in  $\phi$  and thus define *ContainsAtMost*( $\phi, n, \psi$ ).

**Definition 4 (Examples of constraints)** Given a threshold value  $v$ , typical syntactic constraints are *MinLen*( $\phi, v$ )  $\equiv |\phi| \geq v$  and *MaxLen*( $\phi, v$ )  $\equiv |\phi| \leq v$ . Assume that *Supp*( $\phi, r$ ) denotes the number of strings in  $r$  that are super-strings of  $\phi$ , i.e.,  $|\text{ext}(\phi, r)|$ . Given a threshold value  $f$ , *MinSupp*( $\phi, r, f$ )  $\equiv \text{Supp}(\phi, r) \geq f$  (resp. *MaxSupp*( $\phi, r, f$ )  $\equiv \text{Supp}(\phi, r) \leq f$ ) denotes a minimal (resp. maximal) support constraint in  $r$ .

**Example 2** Assume  $r = \{abd, abc, dc, c, dc\}$ , we have *Supp*( $abd, r$ ) = 1, *Supp*( $dc, r$ ) = 2, *Supp*( $ad, r$ ) = 0, and *Supp*( $\epsilon, r$ ) = 5. *MinSupp*( $dc, r, 2$ ), *MaxSupp*( $abd, r, 2$ ), *MoreGeneral*( $c, dc$ ), and *MinLen*( $abd, 3$ ) are examples of satisfied constraints.  $Q \equiv \text{MinSupp}(\phi, r, 2) \wedge \text{MaxSupp}(\phi, r, 4) \wedge \text{MinLen}(\phi, 2)$  is an example of an inductive query whose solution set is  $\{ab, dc\}$ .

The concept of anti-monotonicity and its dual notion of monotonicity is central to our work. When an anti-monotonic constraint like the minimal support is violated by a candidate string, none of its more specific strings (i.e., super-strings) can satisfy it and this gives rise to pruning in the search space. This has been the key property for the many efficient algorithms which mine frequent strings. Negations of anti-monotonic constraints are called monotonic, e.g., the maximal support, and can lead to dual pruning strategies. This has been studied in detail in many papers, e.g., [20, 12].

**Definition 5 ((Anti-)monotonicity)** Let  $r$  be a data set,  $\mathcal{L}$  be the pattern language and  $p$  be parameters. A constraint  $Q$  is anti-monotonic iff  $\forall r$  and  $\forall \phi, \psi \in \mathcal{L}$ ,  $\phi \succeq \psi \Rightarrow Q(\psi, r, p) \rightarrow Q(\phi, r, p)$ . Dually, a constraint  $Q'$  is monotonic iff  $\phi \preceq \psi \Rightarrow Q'(\psi, r, p) \rightarrow Q'(\phi, r, p)$ .

Notice that conjunctions and disjunctions of anti-monotonic (resp. monotonic) constraints are anti-monotonic (resp. monotonic).

**Example 3** *SuperString*( $\phi, \psi$ ), *MinLen*( $\phi, v$ ), and *MaxSupp*( $\phi, r, f$ ) are monotonic constraints. *SubString*( $\phi, \psi$ ), *ContainsAtMost*( $\phi, n, \psi$ ), *MaxLen*( $\phi, v$ ), and *MinSupp*( $\phi, r, f$ ) are anti-monotonic ones.

The evaluation of some constraints on a pattern  $\phi$  does not require to scan  $r$  (e.g., *SuperString*( $\phi, \psi$ ), *MaxLen*( $\phi, v$ )), while to evaluate some others, one needs to find the occurrences of  $\phi$  in  $r$ . For instance, we have defined *MinSupp*( $\phi, r, f$ ) based on a number of strings where  $\phi$  occurs exactly (i.e., the cardinality of  $\{\sigma \in r \text{ such that } \sigma \sqsupseteq \phi\}$ ). However, in many application domains, measures based on such exact occurrences may be misleading. We consider it is important to study a support constraint based on

soft-occurrences. The idea is that a string  $\sigma \in r$  supports  $\phi$  if  $\sigma$  contains a sub-string  $\sigma'$  similar enough to  $\phi$ .  $\sigma'$  is then called a soft-occurrence of  $\phi$ .

Extensive studies of (anti)-monotonicity properties have given rise to efficient search space pruning strategies. It is far more complex and sometime impossible to consider generic algorithms<sup>2</sup> for constraints that do not have the monotonicity properties. An “enumerate and test” strategy is never possible in real-life problems (large alphabets and/or large input sequences and/or huge number of input sequences). A solution might be to heuristically compute part of the solution. We are however convinced that completeness has an invaluable added value, and we prefer to study smart relaxation or decomposition strategies to solve our inductive queries on strings.

**Problem setting.** Our objective is to formalize the concept of soft-support constraints such that they can be processed efficiently, i.e., as combinations of monotonic and anti-monotonic constraints. This will enable to exploit efficient generic strategies for solving arbitrary combinations of soft-support constraints with other (anti)-monotonic constraints [12,13,14]. This is however challenging. Indeed, relevant similarity constraints are generally neither monotonic nor anti-monotonic [18] while our understanding of soft-occurrences relies on similarity constraints. As a result, preserving the (anti)-monotonicity of soft-support constraints can not be guaranteed. Looking for reasonable conditions under which such properties would be preserved is clearly our main technical issue.

## 2. Defining Soft-Occurrences and Soft-Support Constraints

The soft support of a pattern  $\psi$  is derived from a number of its soft-occurrences  $\phi$ , i.e., patterns  $\phi$  such that  $\text{sim}(\phi, \psi)$  where  $\text{sim}$  returns true when the two patterns are similar. It enables to use the similarity approach from [18], slightly modifying the monotonic sub-constraint such that its parameters become less connected to  $|\psi|$ .

**Definition 6 (Longest Common Subsequence)** *Let  $x$  be a pattern from  $\mathcal{L}$ . A subsequence of  $x$  is any string  $w$  that can be obtained from  $x$  by deleting zero or more (not necessarily consecutive) symbols. More formally,  $w$  is a subsequence of  $x$  if there exists integers  $i_1 < i_2 < \dots < i_n$  s.t.  $w_1 = x_{i_1}, w_2 = x_{i_2}, \dots, w_n = x_{i_n}$ .  $w$  is a Longest Common Subsequence (LCS) of  $x$  and  $\phi$  if it is a subsequence of  $x$ , a subsequence of  $\phi$ , and its length is maximal. Notice that  $|w| = \text{lcs}(\phi, x)$  and, in general,  $w$  is not unique.*

**Definition 7 (Insertions, Deletions)** *Let  $x$  be the reference pattern,  $\phi$  be a candidate pattern from  $\mathcal{L}$ . Let fix any LCS of  $\phi$  and  $x$ , and denote the symbols of  $\phi$  (resp.  $x$ ) that do not belong to a LCS as deletions (resp. insertion). The number of deletions (resp. insertions) is  $\text{Dels}(\phi, x) = |\phi| - \text{lcs}(\phi, x)$  (resp.  $\text{Ins}(\phi, x) = |x| - \text{lcs}(\phi, x)$ ). Notice that  $x$  can be produced from  $\phi$  by deleting from  $\phi$  the deletions and inserting into  $\phi$  the insertions.*

**Lemma 1** *Assume  $x, \phi \in \mathcal{L}$ ,  $\phi' \sqsubseteq \phi$ ,  $w$  one LCS of  $\phi$  and  $x$ , and  $w'$  one LCS of  $\phi'$  and  $x$ . We have  $|w| = \text{lcs}(\phi, x) \geq \text{lcs}(\phi', x) = |w'|$ .*

<sup>2</sup>Algorithms not dedicated to a specific combination of primitive constraints

The formal proofs of this lemma and the other propositions or properties are available and can be asked to the authors.

**Definition 8 (Max Insertions constraint)** Let  $x$  be the reference pattern,  $\phi$  be a candidate pattern from  $\mathcal{L}$ , and  $ins$  a threshold value. The Maximum Insertions constraint is defined as  $MaxIns(\phi, x, ins) \equiv Ins(\phi, x) \leq ins$ .

**Proposition 1**  $MaxIns(\phi, x, ins)$  is monotonic.

**Example 4** Assume  $x = cbcddda$ . Patterns  $\phi_1 = dbddda$  and  $\phi_2 = bcddada$  satisfy  $MaxIns(\phi, x, 2)$ :  $Ins(\phi_1, x) = |x| - |bddd| = 2$  and  $Ins(\phi_2, x) = |x| - |bcddda| = 1$ . Pattern  $\phi_3 = accadcdccccddddd$  also satisfies it:  $Ins(\phi_3, x) = |x| - |ccddd| = 2$ .

Constraint  $MaxIns(\phi, x, ins)$  enables to specify a degree of similarity (i.e., a maximum number of non matching symbols on reference), and thus to capture patterns which are similar to the reference one. Note however that  $MinLCS(\phi, x, l)$  does not restrict the dissimilarity of a candidate. Thus, we need for a second constraint that would bound the number of "errors" within a candidate.

**Definition 9 (Max Deletions constraint)** Let  $x$  be the reference pattern,  $\phi$  be a candidate pattern from  $\mathcal{L}$ , and  $dels$  a threshold value. The Maximum Deletions constraint is defined as  $MaxDels(\phi, x, dels) \equiv Dels(\phi, x) \leq dels$ .

**Proposition 2**  $MaxDels(\phi, x, d)$  is anti-monotonic.

**Definition 10 (Similarity constraint)** Given a reference pattern  $x$  and two thresholds  $ins$  and  $dels$ , our similarity constraint for a pattern  $\phi$  w.r.t.  $x$  is defined as  $C_{sim}(\phi, x, ins, dels) \equiv MaxIns(\phi, x, ins) \wedge MaxDels(\phi, x, dels)$ .

**Example 5** Continuing Example 4, patterns  $\phi_1$  and  $\phi_2$  satisfy  $C_{sim}(\phi, x, 2, 1)$ . Pattern  $\phi_4 = dbdddca$  satisfies  $C_{sim}(\phi, x, 2, 2)$  since  $lcs(\phi_4, x) = |x| - |bddd| = 2$ . Pattern  $\phi_3$  does not satisfy neither  $C_{sim}(\phi, x, 2, 1)$  nor  $C_{sim}(\phi, x, 2, 2)$ .

**Definition 11 (Soft-occurrence)** If a string  $\sigma \in r$  contains  $\phi$  s.t.  $C_{sim}(\phi, \psi, ins, dels)$  is satisfied, we say that  $\phi$  is a soft-occurrence of  $\psi$  denoted as  $sOcc(\psi, ins, dels)$ .

Let us now introduce our new support constraints.

**Definition 12 (Soft-support)** If  $sOcc(\phi, ins, dels)_1, \dots, sOcc(\phi, ins, dels)_n$  are the soft-occurrences for  $\phi$  in  $r$ , the soft-support of  $\phi$  (denoted  $SoftSupp(\phi, r, ins, dels)$ ) is  $|ext(sOcc(\phi, ins, dels)_1, r) \cup \dots \cup ext(sOcc(\phi, ins, dels)_n, r)|$ .

**Definition 13 (Minimum/Maximum soft-support)** Given a user-defined threshold  $f$ , the Minimum Soft-support constraint is defined as  $MinSoftSupp(\phi, r, f, ins, dels) \equiv SoftSupp(\phi, r, ins, dels) \geq f$ . The Maximum Soft-support constraint is defined as  $MaxSoftSupp(\phi, r, f, ins, dels) \equiv SoftSupp(\phi, r, ins, dels) \leq f$ .

**Example 6** Continuing Example 2,  $SoftSupp(abd, r, 1, 1) = 2$  and  $\{bd, abc, abd, ab\}$  are the soft-occurrences of  $abd$  on  $r$ .  $SoftSupp(dc, r, 1, 1) = 5$  and  $\{c, dc, db, bc, bd\}$  are the soft-occurrences of pattern  $dc$ . Examples of constraints which are satisfied are  $MinSoftSupp(dc, r, 4, 1, 1)$  and  $MaxSoftSupp(abd, r, 2, 1, 1)$ .

**Table 1.** Support and soft-support

	<i>Supp</i>	$\frac{Supp \times 100\%}{SoftS(1,1)}$	$\frac{Supp \times 100\%}{SoftS(1,2)}$	$\frac{Supp \times 100\%}{SoftS(2,1)}$	$\frac{SoftS(1,1)}{SoftS(1,2)}$	$\frac{SoftS(1,1)}{SoftS(2,1)}$
Mean val	57.89	14.61	12.37	6.9	0.76	0.37
Stand Dev	70.63	21.26	20.6	14.72	0.09	0.18
Min val	23	1.53	1.14	0.54	0.45	0.06
Max val	843	100	100	97.6	1	0.99

**Proposition 3** *Constraint  $MinSoftSupp(\phi, r, ins, dels)$  is anti-monotonic (dually, constraint  $MaxSoftSupp(\phi, r, ins, dels)$  is monotonic) when  $dels \geq ins$ .*

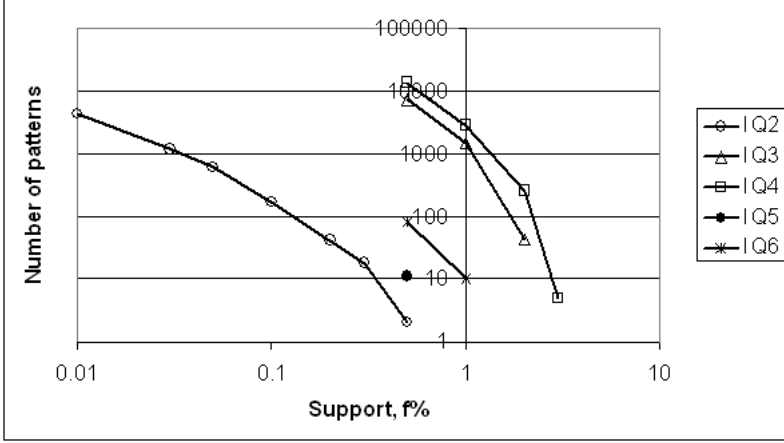
### 3. Experimental Validation

We have performed a number of experiments to empirically evaluate the properties of the  $MinSoftSupp(\phi, r, ins, dels)$  constraint. To the best of our knowledge, FAVST [14] algorithm is among the best algorithms for mining strings that satisfy arbitrary conjunctions of (anti)-monotonic constraints. As a result, the FAVST framework enables to push constraints  $C_{sim}(\phi, x, ins, dels)$ ,  $MinSoftSupp(\phi, r, ins, dels)$  and  $MaxSoftSupp(\phi, r, ins, dels)$  (when  $dels \geq ins$ ), and their arbitrary conjunctions with other anti-(monotonic) constraints. We have developed in C our own implementation of the FAVST algorithm. Our experimental validation has been carried out on the KDD Cup 2000 real-world clickstream datasets [21], using a Intel(R) Pentium(R) M 1.69GHz processor (1GB main memory).

To produce time ordered sequences of templates requested for each session, we have extracted attributes "Session ID", "Request Sequence", "Request Template". There are 137 different request templates, i.e., these sequences are strings over an alphabet of 137 symbols. The produced dataset, referred  $S$ , contains 234,954 strings. The shortest string is of length 1 while the largest one is of length 5,487. We have also extracted the attributes "Session First Request Day Of Week" and "Session First Request Hour Of Day" to split the dataset  $S$  into four datasets: SWE for sessions requested on Saturday or Sunday (47,229 strings), SWD for sessions requested on workdays (187,725 strings), SD for sessions requested from 8 am till 7 pm (137,593 strings), and SN for the sessions requested from 7 pm to 8 am (97,361 strings).

#### 3.1. Comparative Study of Support, Soft-Support and Degrees of Softness

Solving  $MinSoftSupp(\phi, r, ins, dels)$  for  $\phi$  means to solve  $C_{sim}(\psi, \phi, ins, dels)$  to find all patterns  $\psi$  that are soft-occurrences of  $\phi$  given parameters  $ins$  and  $dels$ . We performed experiments to assess the soft-support w.r.t. support, and the impact of different combinations of parameters  $ins$  and  $dels$  on resulting "softness". We have computed  $SoftSupp(\psi, S, 1, 1)$ ,  $SoftSupp(\psi, S, 1, 2)$ , and  $SoftSupp(\psi, S, 2, 1)$  for 796 patterns that are the solutions to  $IQ_1 \equiv MinSupp(\psi, S, 0.01\%) \wedge MinLen(\psi, 7) \wedge MaxLen(\psi, 7)$ . We took the patterns of the same length so that soft support would not be influenced by variable length but only by  $ins$  and  $dels$  values. We got 796 solution patterns. Table 1 provides a statistical summary.



**Figure 1.** Selectivity of  $MinSupp(\phi, r, f)$  and  $MinSoftSupp(\phi, r, sf, ins, dels)$

We observe that, in most cases, the support of a pattern is quite small w.r.t. its soft-support. Also,  $SoftSupp(\psi, S, 1, 1)$  tends to be smaller than  $SoftSupp(\phi, S, 1, 2)$  and  $SoftSupp(\phi, S, 2, 1)$ . Finally,  $SoftSupp(\phi, S, 1, 2)$  tends to be smaller than  $SoftSupp(\phi, S, 2, 1)$ .

### 3.2. Selectivity of Minimal (Soft)-Support Constraints

To compare the selectivity of  $MinSoftSupp(\phi, r, f, ins, dels)$  and  $MinSupp(\phi, r, f)$  constraints we computed solutions to

$$IQ_2 \equiv MinSupp(\phi, S, f) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10),$$

$$IQ_3 \equiv MinSoftSupp(\phi, S, f, 1, 1) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$$

$$IQ_4 \equiv MinSoftSupp(\phi, S, f, 1, 2) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$$

$$IQ_5 \equiv MinSoftSupp(\phi, S, f, 1, 0) \wedge MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$$

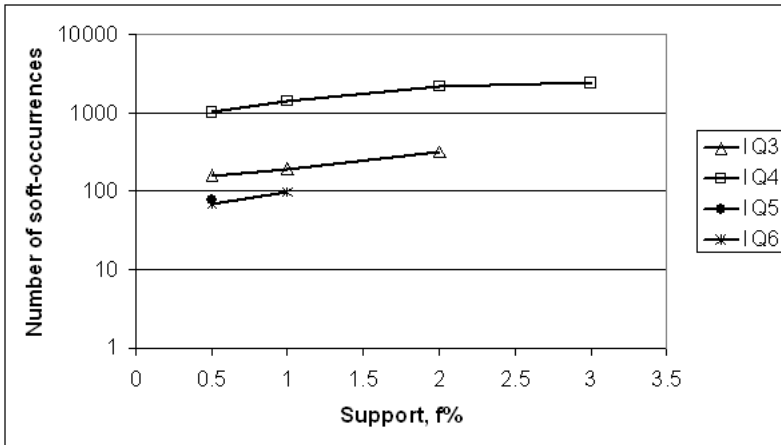
$$IQ_6 \equiv MinSoftSupp(\phi, S, f, 1, 2) \wedge MinLen(\phi, 4) \wedge MaxLen(\phi, 10)$$

The size of the corresponding solutions is plotted against different  $f$  thresholds in the graph given in Figure 1

For  $MinSupp(\phi, S, f)$ , we started at  $f = 0.01\%$ . This is a pretty small value and it appears fair to consider that patterns which do not satisfy this constraint are not interesting. For  $MinSoftSupp(\phi, S, sf, ins, dels)$ , we started at  $f = 0.5\%$  because of consumed time restrictions (see Section 3.4). For both constraints, we increased  $f$  value until the corresponding solution set became empty. In all, there are 767,238 patterns satisfying  $MinLen(\phi, 4) \wedge MaxLen(\phi, 10)$ , and 727,873 patterns satisfying  $MinLen(\phi, 5) \wedge MaxLen(\phi, 10)$ .

Observe that  $MinSupp(\phi, S, f)$  with even very small support thresholds drastically prunes, while the same support values for  $MinSoftSupp(\phi, S, f, ins, dels)$  are not selective at all. It emphasizes the added value for  $MinSoftSupp(\phi, S, f, ins, dels)$ : one might assume that at least 1% of the sessions share common requested templates, and  $MinSoftSupp(\phi, S, 1\%, ins, dels)$  enables to extract these regularities while  $MinSupp(\phi, S, 1\%)$  leads to an empty collection.



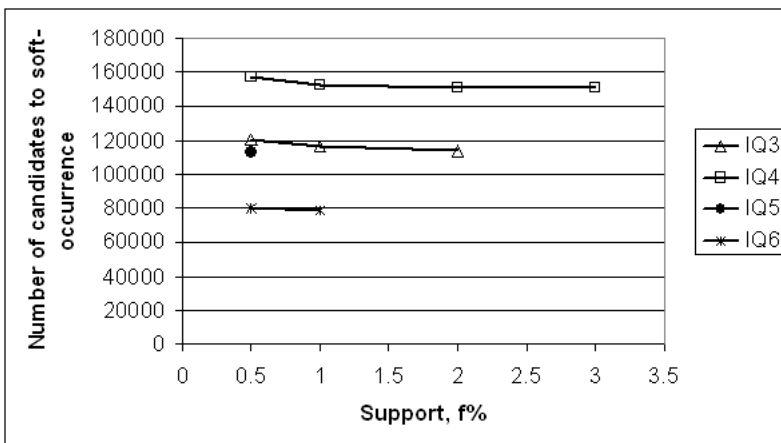


**Figure 2.** Number of soft-occurrences

Figure 2 plots the mean values of the number of soft-occurrences for patterns that are solutions to  $IQ_3$ ,  $IQ_4$ ,  $IQ_5$  and  $IQ_6$ . It reveals that, in general, the greater pattern soft-support the more soft-occurrences, or similar patterns, it has. This is however not a theorem.

### 3.3. Empirical Assessment of Soft-Support Constraint

We strongly believe that softness is needed to find valid regularities or patterns when data or the phenomenon we would like to capture is somehow noisy. We performed experiments to assess this expected added value. In the examples of extracted patterns, i.e., sequences of requested templates, we denote a template by a latin alphabet majuscule letter (see Table 2).



**Figure 3.** Number of candidates to soft-occurrences

**Table 2.** Identifiers of templates

ID	Template
A	main/home\ .jhtm
B	main/departments\ .jhtml
C	main/search_results\ .jhtml
D	products/productDetailLegwear\ .jhtml
E	main/shopping_cart\ .jhtml
F	main/login2\ .jhtml
G	main/registration\ .jhtml
H	main/welcome\ .jhtml
I	checkout/expressCheckout\ .jhtml
J	main/boutique\ .jhtml
K	main/assortment\ .jhtml
L	main/vendor\ .jhtml
M	main/leg_news\ .jhtml
N	products/productDetailLegcare\ .jhtml

We have already mentioned that soft-support constraint helps to identify regularities whose exact-support does not appear discriminant (i.e., using exact support, the relevant pattern is blurred among many other other ones). For instance, solving

$$IQ_7 \equiv \text{MinSoftSupp}(\phi, S, 2\%, 1, 1) \wedge \text{MinLen}(\phi, 5) \wedge \text{MaxLen}(\phi, 10),$$

we retrieve the pattern ABCCD having the highest soft-support (2.9%) among patterns whose length is at least 5. Its exact-support is only 0.18%. As a second example, solving

$$IQ_8 \equiv \text{MinSoftSupp}(\phi, S, 0.5\%, 1, 1) \wedge \text{MinLen}(\phi, 7) \wedge \text{MaxLen}(\phi, 10),$$

we find the pattern DCCCCDD of maximal soft support (0.8%) among the patterns whose length is at least 7. Its exact-support is 0.06% only.

Finally, among the solution patterns to

$$IQ_9 \equiv \text{MinSoftSupp}(\phi, S, 0.5\%, 1, 1) \wedge \text{ContainsAtMost}(\phi, 3, D) \wedge \text{MinLen}(\phi, 7) \wedge \text{MaxLen}(\phi, 10),$$

the maximal soft-support is 0.6% for the pattern CDEFGHI while its exact-support is 0.09%.

To get some empirical feedback on both the soft-support and its corresponding exact-support ratio, we evaluated on the query

$$IQ_{10} \equiv \text{MinSoftSupp}(\phi, S, 0.5\%, 1, 1) \wedge \text{MinLen}(\phi, 5) \wedge \text{MaxLen}(\phi, 10)$$

Table 3 gives the number of patterns for the intervals of the ratio exact-support/soft-support. We observe that the value of exact-support is not discriminant for the major part of the solutions to  $IQ_{10}$ .

Next, we have been looking for sequences of templates that might be specific to workdays (resp. daytime), i.e., frequent among the sessions requested during workdays (resp. daytime) and not frequent among the sessions requested during weekends (resp. nighttime), and vice versa. There were no such sequences of length at least 5 when searching for soft-occurrences with parameters  $ins = 1$  and  $del = 1$ . We thus relaxed the  $\text{MinLen}(\phi, 5)$  constraint and restricted the similarity parameters accordingly. It has given the query

**Table 3.** Number of patterns for exact-support/soft-support intervals

exact-support/soft-support, $r$	Nb of patterns
$0.0002 \leq r < 0.001$	2131
$0.001 \leq r < 0.01$	3159
$0.01 \leq r < 0.1$	1720
$0.1 \leq r < 0.5$	419
$r \geq 0.5$	5

$$IQ_{11} \equiv \text{MinSoftSupp}(\phi, \text{SWD}, 0.6\%, 0, 1) \wedge \text{MaxSoftSupp}(\phi, \text{SWE}, 0.3\%, 0, 1) \wedge \text{MinLen}(\phi, 4) \wedge \text{MaxLen}(\phi, 10).$$

We found 3 patterns in the solution set for  $IQ_{11}$ : JBCC, KBCC, and KKKJ. Similarly, solving

$$IQ_{12} \equiv \text{MinSoftSupp}(\phi, \text{SD}, 0.6\%, 0, 1) \wedge \text{MaxSoftSupp}(\phi, \text{SN}, 0.3\%, 0, 1) \wedge \text{MinLen}(\phi, 4) \wedge \text{MaxLen}(\phi, 10)$$

has given one solution pattern only: DEDE.

In a number of cases, the exact-support and the soft-support of patterns do not coincide. When looking for dataset-characteristic patterns, exact-support can differ significantly, while soft-supports are similar. We extracted patterns satisfying  $\text{MinSupp}(\phi, r_1, f_1) \wedge \text{MaxSupp}(\phi, r_2, f_2)$  constraints and we evaluated their soft-support (with parameters  $ins = 1, dels = 1$  when  $|\phi| \geq 5$ , and  $ins = 0, dels = 1$  otherwise). For instance, the pattern ADDD is a solution pattern to

$$IQ_{13} \equiv \text{MinSupp}(\phi, \text{SWE}, 0.01\%) \wedge \text{MaxSupp}(\phi, \text{SWD}, 0.005\%) \wedge \text{MinLen}(\phi, 4) \wedge \text{MaxLen}(\phi, 10),$$

but its soft-support in SWE is 0.5% and in SWD is 0.4%. Similarly, the pattern AMALA belongs to the solution set of

$$IQ_{14} \equiv \text{MinSupp}(\phi, \text{SN}, 0.1\%) \wedge \text{MaxSupp}(\phi, \text{SD}, 0.05\%) \wedge \text{MinLen}(\phi, 4) \wedge \text{MaxLen}(\phi, 10),$$

but its soft-support in SN is 0.5% and in SD is 0.4%.

Exact-support and soft-support values can be even contradictory. For instance, the patterns DBDBN and ABDBDB belong to the solution set of

$$IQ_{15} \equiv \text{MinSupp}(\phi, \text{SN}, 0.1\%) \wedge \text{MaxSupp}(\phi, \text{SD}, 0.07\%) \wedge \text{MinLen}(\phi, 4) \wedge \text{MaxLen}(\phi, 10),$$

but their soft-supports in SN are 0.3% while they are 0.8% in SD.

These examples emphasize that soft-support constraint are needed to avoid misleading hypothesis on dataset characterization.

### 3.4. Time Efficiency

The runtime to solve  $IQ_2, IQ_3, IQ_4, IQ_5$  and  $IQ_6$  is given in Figure 4. The fact we get a rather poor time efficiency to compute  $\text{MinSoftSupp}(\phi, S, f, ins, dels)$  is not surprising. Firstly,  $\text{MinSoftSupp}(\phi, S, f, ins, dels)$  is far less selective than  $\text{MinSupp}(\phi, S, f)$  (see Figure 1). Then, the evaluation of  $\text{SoftSupp}(\phi, r, ins, dels)$  is much more expensive than exact-support counting: even if we push deeply into the extraction phase the (anti)-monotonic conjuncts  $\text{MaxDels}(\psi, \phi, dels)$  and  $\text{MaxIns}(\psi, \phi, ins)$  (parts of similarity constraint  $C_{sim}(\psi, \phi, ins, dels)$ ), the number of candidates for

which  $C_{sim}(\psi, \phi, ins, dels)$  still has to be evaluated can be huge (e.g., hundreds of thousands, see Figure 3).

The evaluation of  $C_{sim}(\psi, \phi, ins, dels)$  is expensive as well: to compute the Longest Common Subsequence we have employed a classical dynamic programming approach of time complexity  $O(nm)$  [22]. There is clearly a room for improvements here (see, e.g., [23] for a survey). Also, when  $ins = dels$ , one can exploit the symmetric property of the underlying similarity relation. In addition, computations can be tuned by choosing dynamically the order of constraints to push.

Let us notice however that, even though soft-support counting may take hours (as we see in Figure 4), it does not prevent from optimizing sequences of inductive queries involving soft-support constraints. Indeed, it is possible to evaluate once the soft-support for patterns in a dataset for some minimal value  $\alpha$  (resp. maximal value  $\beta$ ) before solving queries which involve  $MinSoftSupp(\phi, r, f, ins, dels)$  for  $f \geq \alpha$  (resp.  $MaxSoftSupp(\phi, r, f, ins, dels)$  for  $f \leq \beta$ ). Also, when computing a conjunction  $MinSoftSupp(\phi, r_1, f_1, ins_1, dels_1) \wedge MaxSoftSupp(\phi, r_2, f_2, ins_2, dels_2)$ , the evaluation of  $MaxSoftSupp(\phi, r_2, f_2, ins_2, dels_2)$  is far less expensive since we prune its search space by the anti-monotonic constraint  $MinSoftSupp(\phi, r_1, f_1, ins_1, dels_1)$  and compute  $MaxSoftSupp(\phi, r_2, f_2, ins_2, dels_2)$  by starting at the  $S$  border for the anti-monotonic constraint  $MinSoftSupp(\phi, r_1, f_1, ins_1, dels_1)$  (see, e.g., [24]), ascending towards more general patterns. Figure 5 provides the needed time for solving  $MaxSoftSupp(\phi, SN, \beta, ins, dels)$  when  $1\% \leq \beta \leq 5\%$  in a conjunction  $MinSoftSupp(\phi, SD, \alpha = 0.5\%, ins, dels) \wedge MaxSoftSupp(\phi, SN, \beta, ins, dels)$  for pairs of parameters  $ins = 0, dels = 1$  and  $ins = 1, dels = 1$ . Notice also that we can store the computed patterns and their soft-supports such that the future extractions can be drastically optimized.

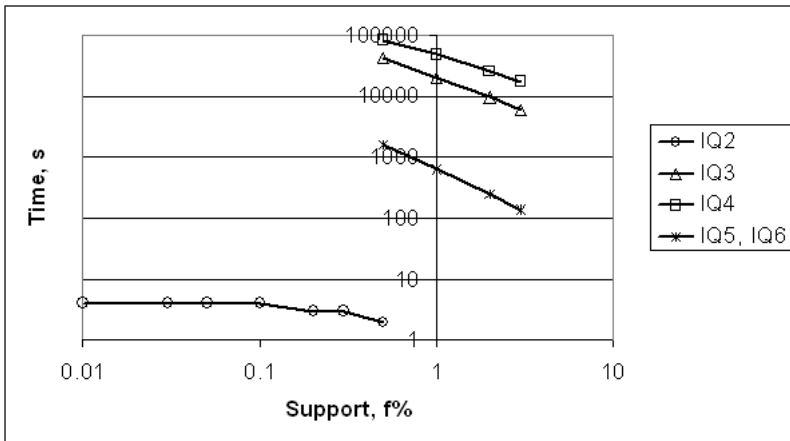


Figure 4. Time efficiency

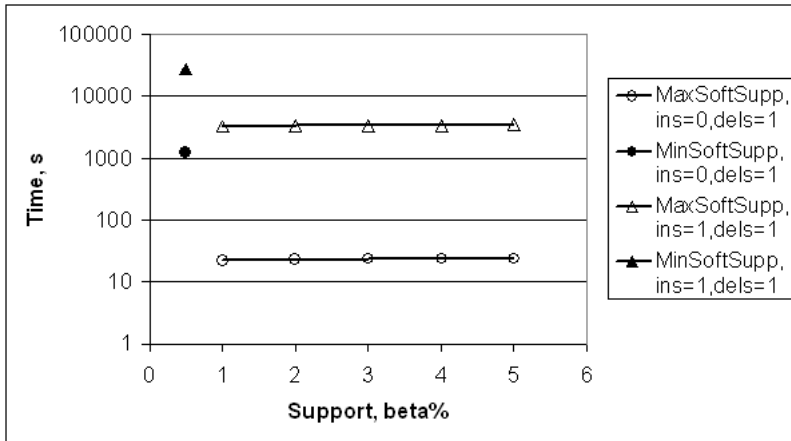


Figure 5. Time efficiency for  $MaxSoftSupp(\phi, SN, \beta, ins, dels)$

#### 4. Conclusion

The vision of the inductive database framework is that expert data owners might be able to query both the data and the patterns holding in the data. In this paper, we have considered the so-called inductive querying problem on string datasets, i.e., the evaluation of constraints which specify declaratively the desired properties for string patterns. Solving arbitrary combinations of useful primitive constraints by means of generic algorithms is challenging. In this paper, we revisited the popular support constraints when introducing soft occurrences. It might be quite useful when dealing with intrinsically noisy data sets. We formalized an approach to soft-support constraint checking which can take the most from efficient strategies for solving conjunctions of monotonic and anti-monotonicity constraints. As a result, the analysts can combine our soft-support constraints with many other user-defined constraints of interest.

#### Acknowledgements

We wish to thank Blue Martini Software for contributing the KDD Cup 2000 data. This research is partly funded by ACI MD 46 Bingo (French government) and by EU contract IST-FET IQ FP6-516169.

#### References

- [1] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *CACM*, 39(11):58–64, 1996.
- [2] L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2003.
- [3] J.-F. Boulicaut. Inductive databases and multiple uses of frequent itemsets: the cInQ approach. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, pages 1–23. Springer-Verlag LNCS 2682, 2004.
- [4] J.-F. Boulicaut, L. De Raedt, and H. Mannila (Eds.). *Constraint-based Mining and Inductive Databases*. Springer-Verlag LNAI 3848, 2006.

- [5] M. Botta, J.-F. Boulicaut, C. Masson, and R. Meo. Query languages supporting descriptive rule mining: a comparative study. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, pages 27–56. Springer-Verlag LNCS 2682, 2004.
- [6] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings EDBT '96*, pages 3–17. Springer-Verlag, 1996.
- [7] F. Massegli, F. Cathala, and P. Poncelet. The PSP approach for mining sequential patterns. In *Proceedings PKDD '98*, pages 176–184. Springer-Verlag, 1998.
- [8] M.J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1-2):31–60, 2001.
- [9] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings ACM SIGKDD '00*, pages 355–359. ACM Press, 2000.
- [10] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings IEEE ICDE '01*, pages 215–224. IEEE Computer Society, 2001.
- [11] M. Leleu, C. Rigotti, J.-F. Boulicaut, and G. Euvrard. Constraint-based mining of sequential patterns over datasets with consecutive repetitions. In *Proceedings PKDD 2003*, pages 303–314. Springer-Verlag, 2003.
- [12] L. De Raedt, M. Jaeger, S. Dan Lee, and H. Mannila. A theory of inductive query answering. In *Proceedings IEEE ICDM '02*, pages 123–130. IEEE Computer Society, 2002.
- [13] S. Dan Lee and L. De Raedt. An algebra for inductive query evaluation. In *Proceedings IEEE ICDM '03*, pages 147–154. IEEE Computer Society, 2003.
- [14] S. Dan Lee and L. De Raedt. An efficient algorithm for mining string databases under constraints. In *Proceedings KDD '04*, pages 108–129. Springer-Verlag, 2004.
- [15] M. N. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. In *Proceedings VLDB '99*, pages 223–234. Morgan Kaufmann Publishers Inc., 1999.
- [16] J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *Proceedings ACM CIKM '02*, pages 18–25. ACM Press, 2002.
- [17] H. Albert-Lorincz and J.-F. Boulicaut. Mining frequent sequential patterns under regular expressions: a highly adaptive strategy for pushing constraints. In *Proceedings SIAM DM 2003*, pages 316–320. SIAM, 2003.
- [18] I. Mitasiunaite and J.-F. Boulicaut. Looking for monotonicity properties of a similarity constraint on sequences. In *Proceedings of ACM SAC '06, Special Track on Data Mining*, pages 546–552. ACM Press, 2006.
- [19] I. Mitasiunaite and J.-F. Boulicaut. About softness for inductive querying on sequence databases. In *Proceedings of DB&IS '06*, pages 77–82. IEEE, 2006.
- [20] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
- [21] R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86–98, 2000. <http://www.ecn.purdue.edu/KDDCUP>.
- [22] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *CACM*, 18(6):341–343, 1975.
- [23] A. Apostolico. String editing and longest common subsequences. In *Handbook of Formal Languages*, volume 2 Linear Modeling: Background and Application, pages 361–398. Springer-Verlag, 1997.
- [24] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, (18):203–226, 1981.

# Data Engineering

This page intentionally left blank



# A Multiple Correspondence Analysis to Organize Data Cubes

Riadh BEN MESSAOUD, Omar BOUSSAID and Sabine LOUDCHER RABASÉDA

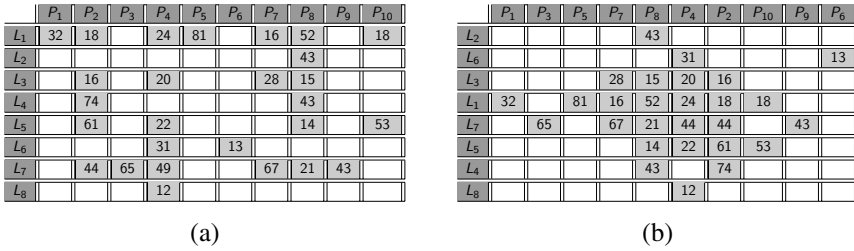
*Laboratory ERIC – University of Lyon 2  
5 avenue Pierre Mendès-France, 69676, Bron Cedex – France*

**Abstract.** On Line Analytical Processing (OLAP) is a technology basically created to provide users with tools in order to explore and navigate into data cubes. Unfortunately, in huge and sparse data, exploration becomes a tedious task and the simple user's intuition or experience does not lead to efficient results. In this paper, we propose to exploit the results of the Multiple Correspondence Analysis (MCA) in order to enhance data cube representations and make them more suitable for visualization and thus, easier to analyze. Our approach addresses the issues of organizing data in an interesting way and detects relevant facts. Our purpose is to help the interpretation of multidimensional data by efficient and simple visual effects. To validate our approach, we compute its efficiency by measuring the quality of resulting multidimensional data representations. In order to do so, we propose an homogeneity criterion to measure the visual relevance of data representations. This criterion is based on the concept of geometric neighborhood and similarity between cells. Experimental results on real data have shown the interest of using our approach on sparse data cubes.

**Keywords.** OLAP, data cubes, data representation, MCA, test-values, arrangement of attributes, characteristic attributes, homogeneity criterion

## Introduction

On-Line Analytical Processing (OLAP) is a technology supported by most data warehousing systems [1,2]. It provides a platform for analyzing data according to multiple dimensions and multiple hierarchical levels. Data are presented in multidimensional views, commonly called data cubes [3]. A data cube can be considered as a space representation composed by a set of cells. A cell is associated with one or more measures and identified by coordinates represented by one attribute from each dimension. Each cell in a cube represents a precise fact. For example, if dimensions are *products*, *stores* and *months*, the measure of a particular cell can be the *sales* of one *product* in a particular *store* on a given *month*. OLAP provides the user with visual based tools to summarize, explore and navigate into data cubes in order to detect interesting and relevant information. However, exploring a data cube is not always an easy task to perform. Obviously, in large cubes containing sparse data, the whole analysis process becomes tedious and complex. In such a case, an intuitive exploration based on the user's experience does not quickly lead to efficient results. More generally, in the case of a data cube with more than three dimensions, a user is naturally faced to a hard task of navigation and exploration in order



**Figure 1.** Example of two representations of a 2-dimensional data cube.

to detect relevant information. Current OLAP provides query-driven and visual tools to browse data cubes, but does not deeply assist the user and help him/her to investigate interesting patterns.

For example, consider the cube of Figure 1. On the one hand, representation 1 (a) displays sales of products ( $P_1, \dots, P_{10}$ ) crossed by geographic locations of stores ( $L_1, \dots, L_8$ ). In this representation, full cells (gray cells) are displayed randomly according to the lexical ordering of the *attributes* – also called *members* – in each dimension. The way the cube is displayed does not provide an attractive representation that visually helps a user to easily interpret data.

On the other hand, Figure 1 (b) contains the same information as Figure 1 (a). However, it displays a data representation which is visually easier to analyze. In fact, the cube of Figure 1 (b) expresses important relationships by providing a visual representation that gathers full cells together and separates them from empty ones. In a natural way, such a representation is more comfortable to the user and allows to drive easy and efficient analysis.

Nevertheless, note that the representation (b) of Figure 1 can be interactively constructed by the user from representation (a) via some classic OLAP operators. This suppose that the user intuitively knows how to arrange the attributes of each dimension. Hence, we propose to provide the user with an automatic assistance to identify interesting facts and arrange them in a suitable visual representation [4,5]. As shown in Figure 1, we propose an approach that allows the user to get relevant facts expressing relationships and displays them in an appropriate way that enhances the exploration process independently of the cube's size. Thus, we suggest to carry out a Multiple Correspondence Analysis [6] (MCA) on a data cube as a preprocessing step. Basically, MCA is a powerful describing method even for huge volumes of data. It factors categorical variables and displays data in a factorial space constructed by orthogonal system of axes that provides relevant views of data. These elements motivate us to exploit the results of the MCA in order to better explore large data cubes by identifying and arranging its interesting facts. The first constructed factorial axis summarizes the maximum of information contained in the cube. We focus on relevant OLAP facts associated with characteristic attributes (variables) given by the factorial axes. These facts are interesting since they reflect relationships and concentrate a significant information. For a better visualization of these facts, we highlight them and arrange their attributes in the data space representation by using the *test-values* [7].

In order to evaluate the visual relevance of multidimensional data representations, we also propose in this paper a novel criterion to measure the homogeneity of cells

distribution in the space representation of a data cube [8]. This criterion is based on geometric neighborhood of data cube cells, and also takes into account the similarity of cells' measures and provides a scalar quantification for the homogeneity of a given data cube representation. It also allows to evaluate the performance of our approach by comparing the quality of the initial data representation and the arranged one.

This paper is organized as follows. In section 1, we present related work to our approach. We provide in section 2 the problem formalization and present the general context of this work. The section 4 introduces the *test-values* and details the steps of our approach. We define in the next section our quality representation criterion. The section 6 presents a real world case study on a huge and sparse data cube. We propose experimental results in the section 7. Finally, we conclude and propose some future researches directions.

## 1. Related Work

Several works have already treated the issue of enhancing the space representation of data cubes. These works were undertaken following different motivations and adopted different ways to address the problem. We note that while many efforts are interested to computational aspects of data cubes (optimization of storage space, compression strategies, queries response time, etc.), a small number of studies have focused on OLAP aspects. Our present work fits into the second category. In our work, we focus on assisting OLAP users in order to improve and help the analysis process on large and sparse data cubes. We use a factorial approach to highlight relevant facts and provide an interesting visual data representations. Nevertheless, we dress an overview of main studies as well in the first as in the second category.

Some studies approximate computation of compressed data cube. In [9], Vitter *et al.* proposed to build compact data cubes by using approximation through wavelets. Quasi-Cube [10] compresses data representation by materializing only sufficient parts of a data cube. In [11] approximation is performed by estimating the density function of data. Other efforts address the issue of computing data cubes with index structure. For instance, Dwarf [12] uses indexes to reduce the storage space of a cube by identifying and factoring redundant tuples. Wang *et al.* propose to factorize data redundancies with BST [13] (*Base Single Tuple*). In [14], Feng *et al.* introduce PrefixCube, a data structure based on only one BST. The Quotient Cube [15] summarizes the semantic contents of a data cube and partitions it into similar cells. In [16], QC-Tree is directly constructed from the base table in order to maintain it under updates. Some other studies optimize storage spaces by partitioning the initial cube. Range CUBE [17] identifies correlations between attributes and compresses the data cube. Partitioned-Cube [18] partitions large relations into fragments. Operations on the cube are, therefore, performed in memory-sized fragments independently. In [19], high dimensional data are transformed into small local cubes and used to for online queries.

Finally, our approach shares already the same motivation of Choong *et al.* [20]. The authors address the problem of high dimensionality of data cubes. They try to enhance analysis processes by preparing the dataset into appropriate representation. Thus, the user can explore it in a more effective manner. The authors use an approach that combines association rules algorithm and a fuzzy subsets method. Their approach consists in

identifying blocks of similar measures in the data cube. However, this approach does not take into account the problem of data sparsity. Furthermore, it does not provide a quality evaluation of the resulting multidimensional representations.

We emphasize that our approach does not deal with the issues of data cube compression, reduction of dimensionality or optimization of storage space. Through this study, we try to act on sparsity in huge multidimensional representations. Not to reduce it, but to reduce its negative effects on the interpretations and OLAP analysis of data [4,5]. Thus, we use the MCA to arrange differently the facts and highlight their relevant relationships in a data cube within a visual effect that gathers them as well as possible in the space representation.

## 2. Problem Formalization

Let  $\mathcal{C}$  denote a data cube. Note that, our approach can be applied directly on  $\mathcal{C}$  or on a data view (a sub-cube) extracted from  $\mathcal{C}$ . It is up to the user to select dimensions, fix one hierarchical level per dimension and select measures in order to create a particular data view (s)he wishes to visualize. Thus, to enhance the data representation of the constructed view, the user can apply on it our proposed approach. In order to lighten the formalization, in the followings of the paper, we assume that a user has selected a data cube  $\mathcal{C}$ , with  $d$  dimensions  $(D_1, \dots, D_t, \dots, D_d)$ ,  $m$  measures  $(M_1, \dots, M_q, \dots, M_m)$  and  $n$  facts. We also assume that the user has fixed one hierarchical level with  $p_t$  categorical attributes per dimension. Let  $a_j^t$  the  $j^{th}$  attribute of the dimension  $D_t$  and  $p = \sum_{t=1}^d p_t$  the total number of attributes in  $\mathcal{C}$ . For each dimension  $D_t$ , we note  $\{a_1^t, \dots, a_j^t, \dots, a_{p_t}^t\}$  the set of its attributes.

In a first step, the aim of our approach is to organize the space representation of a given data cube  $\mathcal{C}$  by arranging the attributes of its dimensions. For each dimension  $D_t$ , our approach establishes a new arrangement of its attributes  $a_j^t$  in the data space (see subsection 4.2). This arrangement provides a data representation visually easier to interpret and displays multidimensional information in a more suitable way for analysis. In a second step, our approach detects from the resulted representation relevant facts expressing interesting relationships. To do that, we select from each dimension  $D_t$  a subset  $\Phi_t$  of significant attributes, also called characteristic attributes (see subsection 4.3). The crossing of these particular attributes allows to identify relevant cells in the cube.

Our approach is based on the MCA [6,7]. The MCA is a factorial method that displays categorical variables in a property space which maps their associations in two or more dimensions. From a table of  $n$  observations and  $p$  categorical variables, describing a  $p$ -dimensional cloud of individuals ( $p < n$ ), the MCA provides orthogonal axes to describe the most variance of the whole data cloud. The fundamental idea is to reduce the dimensionality of the original data thanks to a reduced number of variables (factors) which are a combination of the original ones. The MCA is generally used as an exploratory approach to unearth empirical regularities of a dataset.

In our case, we assume the cube's facts as the individuals of the MCA, the cube's dimensions as its variables, and the attributes of a dimension as values of their corresponding variables. We apply the MCA on the  $n$  facts of the cube  $\mathcal{C}$  and use its results to build *test-values* (see subsection 4.1) for the attributes  $a_j^t$  of the dimensions  $D_t$ . We exploit these *test-values* to arrange attributes and detect characteristic ones in their corresponding dimensions.

Id	$D_1$	$D_2$	$D_3$	$M_1$	$Z$						
					$Z_1$		$Z_2$		$Z_3$		
Id	$L_1$	$T_2$	$P_1$		$L_1$	$L_2$	$T_1$	$T_2$	$P_1$	$P_2$	$P_3$
1	$L_1$	$T_2$	$P_1$	9	1	0	0	1	1	0	0
2	$L_2$	$T_2$	$P_3$	5	0	1	0	1	0	0	1
3	$L_2$	$T_1$	$P_2$	6	0	1	1	0	0	1	0
4	$L_1$	$T_1$	$P_3$	7	1	0	1	0	0	0	1

(a)
(b)

**Figure 2.** Example of a conversion of a data cube to a complete disjunctive table.

### 3. Applying the MCA on a Data Cube

Like all statistical methods, MCA needs a tabular representation of data as input. Therefore, we can not apply it directly on multidimensional representations like data cubes. Therefore, we need to convert  $\mathcal{C}$  to a *complete disjunctive table*. For each dimension  $D_t$ , we generate a binary matrix  $Z_t$  with  $n$  rows and  $p_t$  columns. Rows represent facts, and columns represent dimension's attributes. The  $i^{th}$  row of  $Z_t$  contains  $(p_t - 1)$  times the value 0 and one time the value 1 in the column that fits with the attribute taken by the fact  $i$ . The general term of  $Z_t$  is:

$$z_{ij}^t = \begin{cases} 1 & \text{if the fact } i \text{ takes the attribute } a_j^t \\ 0 & \text{else} \end{cases} \quad (1)$$

By merging the  $d$  matrices  $Z_t$ , we obtain a complete disjunctive table  $Z = [Z_1, Z_2, \dots, Z_t, \dots, Z_d]$  with  $n$  rows and  $p$  columns. It describes the  $d$  positions of the  $n$  facts of  $\mathcal{C}$  through a binary coding. For instance, Figure 2 shows an simple example of a data cube (a), with 3 dimensions  $D_1 : \{L_1, L_2\}$ ,  $D_2 : \{T_1, T_2\}$ , and  $D_3 : \{P_1, P_2, P_3\}$ . This cube is converted to a complete disjunctive table  $Z$  in Figure 2 (b). In the case of a large data cube, we naturally obtain a very huge matrix  $Z$ . Recall that MCA is a factorial method perfectly suited to huge input dataset with high numbers of rows and columns.

Once the complete disjunctive table  $Z$  is built, MCA starts by constructing a matrix  $B = Z'Z$  – called *Burt table* –, where  $Z'$  is the transposed matrix of  $Z$ . *Burt table*  $B$  is a  $(p, p)$  symmetric matrix which contains all the category marginal on the main diagonal and all possible cross-tables of the  $d$  dimensions of  $\mathcal{C}$  in the off-diagonal. Let  $X$  be a  $(p, p)$  diagonal matrix which has the same diagonal elements of  $B$  and zeros otherwise. We construct from  $Z$  and  $X$  a new matrix  $S$  according to the formula:

$$S = \frac{1}{d} Z'ZX^{-1} = \frac{1}{d} BX^{-1} \quad (2)$$

By diagonalizing  $S$ , we obtain  $(p - d)$  diagonal elements, called *eigenvalues* and denoted  $\lambda_\alpha$ . Each eigenvalue  $\lambda_\alpha$  is associated to a directory vector  $u_\alpha$  and corresponds to a factorial axis  $F_\alpha$ , where  $Su_\alpha = \lambda_\alpha u_\alpha$ .

An eigenvalue represents the amount of inertia (variance) that reflects the relative importance of its axis. The first axis always explains the most inertia and has the largest eigenvalue. Usually, in a factorial analysis process, researchers keep only the first, two

or three axes of inertia. Other researchers give complex mathematical criterion [21,22, 23,24] to determine the number of axes to keep. In [6], Benzecri suggests that this limit should be fixed by user's capacity to give a meaningful interpretation to the axes he keeps. It is not because an axis has a relatively small eigenvalue that we should discard it. It can often help to make a fine point about the data. It is up to the user to choose the number  $k$  of axis to keep by checking eigenvalues and the general meaning of axes.

#### 4. Organizing Data Cubes and Detecting Relevant Facts

Usually in a factorial analysis, relative contributions of variables are used to give sense to the axes. A relative contribution shows the percentage of inertia of a particular axis which is explained by an attribute. The largest relative contribution of a variable to an axis is, the more it gives sense to this axis. In our approach, we interpret a factorial axis by characteristic attributes detected through the use of the *test-values* proposed by Lebart *et al.* in [7]. In the followings, we present the theoretical principle of test-values applied to the context of our approach.

##### 4.1. Test-Values

Let  $I(a_j^t)$  denotes the set of facts having  $a_j^t$  as attribute in the dimension  $D_t$ . We also note  $n_j^t = \text{Card}(I(a_j^t)) = \sum_{i=1}^n z_{ij}^t$  the number of elements in  $I(a_j^t)$ . It corresponds to the number of facts in  $\mathcal{C}$  having  $a_j^t$  as attribute (weight of  $a_j^t$  in the cube).  $\varphi_{\alpha j}^t = \frac{1}{n_j^t \sqrt{\lambda_\alpha}} \sum_{i \in I(a_j^t)} \psi_{\alpha i}$  is the coordinate of  $a_j^t$  on the factorial axis  $F_\alpha$ , where  $\psi_{\alpha i}$  is the coordinate of the facts  $i$  on  $F_\alpha$ . Suppose that, under a null hypothesis  $H_0$ , the  $n_j^t$  facts are selected randomly in the set of the  $n$  facts, the mean of their coordinates in  $F_\alpha$  can be represented by a random variable  $Y_{\alpha j}^t = \frac{1}{n_j^t} \sum_{i \in I(a_j^t)} \psi_{\alpha i}$ , where  $E(Y_{\alpha j}^t) = 0$  and  $\text{VAR}_{H_0}(Y_{\alpha j}^t) = \frac{n-n_j^t}{n-1} \frac{\lambda_\alpha}{n_j^t}$ .

Remark that  $\varphi_{\alpha j}^t = \frac{1}{\sqrt{\lambda_\alpha}} Y_{\alpha j}^t$ . Thus,  $E(\varphi_{\alpha j}^t) = 0$ , and  $\text{VAR}_{H_0}(\varphi_{\alpha j}^t) = \frac{n-n_j^t}{n-1} \frac{1}{n_j^t}$ . Therefore, the test-value of the attribute  $a_j^t$  is:

$$V_{\alpha j}^t = \sqrt{n_j^t \frac{n-1}{n-n_j^t}} \varphi_{\alpha j}^t \quad (3)$$

$V_{\alpha j}^t$  measures the number of standard deviations between the attribute  $a_j^t$  (the gravity center of the  $n_j^t$  facts) and the center of the factorial axis  $F_\alpha$ . The position of an attribute is interesting for a given axis  $F_\alpha$  if its cloud of facts is located in a narrow zone in the direction  $\alpha$ . This zone should also be as far as possible from the center of the axis. The test-value is a criterion that quickly provides an appreciation if an attribute has a *significant* position on a given factorial axis or not.

##### 4.2. Arrangement of Attributes

In a classic OLAP representation of data cubes, attributes are usually organized according to a lexical order such as alphabetic order for *geographic* dimensions or chronological

order for *times* dimensions. In our approach, we propose to exploit the test-values of attributes in order to organize differently the data cube's facts. The new organization will display a relevant data representation easier to analyze and to interpret, especially in the case of large and sparse cubes. For each dimension, we sort its attributes according to the increasing order of their test-values. Actually, a test-value indicates the position of an attribute on a given axis. The relative geometric position of an attribute is more significant to factorial axes when these axes are important (have the greatest eigenvalues). For this, we propose to sort attributes according to the  $k$  first axes selected by the user. We sort the  $p_t$  test-values  $V_{\alpha j}^t$  of the attributes  $a_j^t$  on the axis  $F_\alpha$ . This will provide a new order of indices  $j$ . According to this order, we arrange attributes  $a_j^t$  in the dimension  $D_t$ .

In general, we assume that all attributes of a dimension  $D_t$  are geometrically ordered in the data cube space representation according to the order of indices  $j_t$ . i.e, the attribute  $a_{j_t-1}^t$  precedes  $a_{j_t}^t$  and  $a_{j_t}^t$  precedes  $a_{j_t+1}^t$  (see the example of Figure 3). Indices  $j_t$  are ordered according to the arrangement of the attributes in the space representation of the dimension  $D_t$ .

#### 4.3. Characteristic Attributes

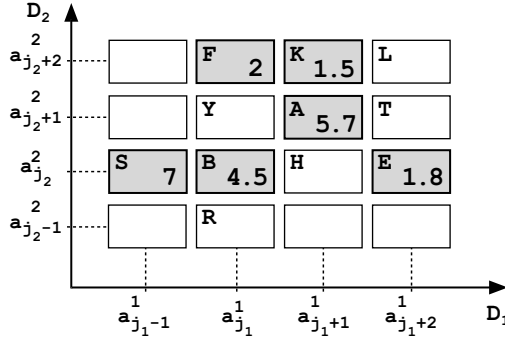
In general, an attribute is considered significant for an axis if the absolute value of its test-value is higher than  $\tau = 2$ . This roughly corresponds to an error threshold of 5%. We note that, the lower error threshold is, the greater  $\tau$  is. In our case, for one attribute, the test of the hypothesis  $H_0$  can induce a possible error. This error will inevitably be increased when we perform the test  $p$  times for all the attributes of the cube. To minimize this accumulation of errors, we propose to fix for each test an error threshold of 1% which correspond to  $\tau = 3$ . We also note that, when a given axis can be characterized by too much attributes according to their test-values, instead of taking them all, we can restrict the selection by considering only a percentage of the most characteristic ones. i.e, those having the highest absolute test-values. Finally to detect interesting facts in a data cube, for each dimension  $D_t$ , we select the following set of characteristic attributes.

$$\Phi_t = \left\{ a_j^t, \text{ where } \forall j \in \{1, \dots, p_t\}, \right. \\ \left. \exists \alpha \in \{1, \dots, k\} \text{ such as } |V_{\alpha j}^t| \geq 3 \right\} \quad (4)$$

### 5. Quality of a Data Representation

We provide a quality criterion of data cube representations [8]. It measures the homogeneity of the geometric distribution of cells in a data cube. One cell contains one or more measures of an OLAP fact. The attributes of a cell are coordinates of a fact according to dimensions in the data space representation. Let  $A = (a_{j_1}^1, \dots, a_{j_t}^t, \dots, a_{j_d}^d)$  be a cell in  $\mathcal{C}$ , where  $t \in \{1, \dots, d\}$  and  $j_t \in \{1, \dots, p_t\}$ .  $j_t$  is the index of the attribute that takes the cell  $A$  according to  $D_t$ . We denote  $|A|$  the value of a measure contained in  $A$  which is equal to NULL if  $A$  is empty. For example, in Figure 3,  $|A| = 5.7$  whereas  $|Y| = \text{NULL}$ . We say that a cell  $B = (b_{j_1}^1, \dots, b_{j_t}^t, \dots, b_{j_d}^d)$  is neighbor of  $A$ , denoted  $B \dashv A$ , if  $\forall t \in \{1, \dots, d\}$ , the coordinates of  $B$  satisfy:

$$b_{j_t}^t = a_{j_t-1}^t, \text{ or } b_{j_t}^t = a_{j_t}^t, \text{ or } b_{j_t}^t = a_{j_t+1}^t$$



**Figure 3.** A 2-dimensional example of a data cube.

In Figure 3, cell  $B$  is neighbor of  $A$  ( $B \dashv A$ ).  $Y$  is also neighbor of  $A$  ( $Y \dashv A$ ). Whereas cells  $S$  and  $R$  are not neighbors of  $A$ . For a cell  $A$  of a cube  $\mathcal{C}$ , we define the neighborhood of  $A$ , denoted  $\mathcal{N}(A)$ , by the set of all cells  $B$  of  $\mathcal{C}$  neighbors of  $A$ .

$$\mathcal{N}(A) = \{B \in \mathcal{C} \text{ where } B \dashv A\}$$

For example, in Figure 3, the neighborhood of  $A$  corresponds to the set  $\mathcal{N}(A) = \{F, K, L, T, E, H, B, Y\}$ . To evaluate similarities between neighbor cells, we define a similarity function  $\delta$ .

**Definition** The similarity  $\delta$  of two cells  $A$  and  $B$  from a cube  $\mathcal{C}$  is defined as follows:

$$\delta : \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{R}$$

$$\delta(A, B) \longmapsto \begin{cases} 1 - \left( \frac{||A| - |B||}{\max(\mathcal{C}) - \min(\mathcal{C})} \right) & \text{if } A \text{ and } B \text{ are full} \\ 0 & \text{else} \end{cases}$$

Where  $||A| - |B||$  is the absolute difference of measures contained in cells  $A$  and  $B$ , and  $\max(\mathcal{C})$  (respectively,  $\min(\mathcal{C})$ ) is the maximum (respectively, the minimum) measure value in cube  $\mathcal{C}$ .

In the cube of Figure 3, where grayed cells are full and white ones are empty,  $\max(\mathcal{C}) = 7$ , which matches with the cell  $S$  and  $\min(\mathcal{C}) = 1.5$ , which matches with the cell  $K$ . For instance,  $\delta(A, B) = 1 - \left( \frac{|5.7 - 4.5|}{7 - 1.5} \right) \simeq 0.78$  and  $\delta(A, Y) = 0$ .

Now, let consider a function  $\Delta$  from  $\mathcal{C}$  to  $\mathbb{R}$  such as  $\forall A \in \mathcal{C}, \Delta(A) = \sum_{B \in \mathcal{N}(A)} \delta(A, B)$ . It corresponds to the sum of the similarities of  $A$  with all its full neighbor cells. For instance, according to Figure 3,  $\Delta(A)$  is computed as follows:  $\Delta(A) = \delta(A, F) + \delta(A, K) + \delta(A, L) + \delta(A, T) + \delta(A, E) + \delta(A, H) + \delta(A, B) + \delta(A, Y) \simeq 1.64$ .

We introduce the crude homogeneity criterion of a data cube  $\mathcal{C}$  according to:

$$chc(\mathcal{C}) = \sum_{\substack{A \in \mathcal{C} \\ |A| \neq \text{NULL}}} \sum_{B \in \mathcal{N}(A)} \delta(A, B) = \sum_{\substack{A \in \mathcal{C} \\ |A| \neq \text{NULL}}} \Delta(A)$$

The crude homogeneity criterion computes the sum of similarities of every couple of full and neighbor cells in a data cube. For instance, in Figure 3, the crude homogeneity



criterion is computed as  $chc(\mathcal{C}) = \Delta(F) + \Delta(K) + \Delta(A) + \Delta(S) + \Delta(B) + \Delta(E) \simeq 6.67$ . Note that, the crude homogeneity criterion of a data cube touches its maximum when all the cells of the cube are full and have equal measures. We denote  $chc_{max}(\mathcal{C}) = \sum_{A \in \mathcal{C}} \sum_{B \in \mathcal{N}(A)} 1$ .

**Definition** The homogeneity criterion of a data cube is defined as:

$$hc(\mathcal{C}) = \frac{chc(\mathcal{C})}{chc_{max}(\mathcal{C})} = \frac{\sum_{\substack{A \in \mathcal{C} \\ |A| \neq \text{NULL}}} \Delta(A)}{\sum_{A \in \mathcal{C}} \sum_{B \in \mathcal{N}(A)} 1}$$

The homogeneity criterion evaluates the quality of a multidimensional data representation. This quality is rather better when full and similar cells are neighbors. Indeed, when similar cells are gathered in specific regions of the space representation of a data cube, this cube is easier to visualize and so, a user can directly focus his/her data interpretation on these regions.

For example, in Figure 3,  $chc_{max}(\mathcal{C}) = 84$ . So, the homogeneity criterion of this representation is:  $hc(\mathcal{C}) = \frac{6.67}{84} \simeq 0.08$ . Nevertheless, such a criterion can not make real sense for a single situation of a data representation. In all cases, we should rather compare it to other data representations of the same cube. In fact, recall that the aim of our method is to organize the facts of an initial data cube representation by arranging attributes in each dimensions according to the order of test-values. Let us denote the initial cube  $\mathcal{C}_{ini}$  and the organized one  $\mathcal{C}_{org}$ . To measure the relevance of the organization provided by our method, we compute the gain  $g = \frac{hc(\mathcal{C}_{org}) - hc(\mathcal{C}_{ini})}{hc(\mathcal{C}_{ini})}$  realized by the homogeneity criterion.

We also note that, for the same cube, its organized representation does not depend on the initial representation because the results of the MCA are insensitive to the order of input variables.

## 6. A Case Study

To test and validate our approach, we apply it on a 5-dimensional cube ( $d = 5$ ) that we have constructed from the *Census-Income Database*<sup>1</sup> of the *UCI Knowledge Discovery in Databases Archive*<sup>2</sup>. This data set contains weighted census data extracted from the 1994 and 1995 current population surveys conducted by the *U.S. Census Bureau*. The data contains demographic and employment related variables. The constructed cube contains 199 523 facts and one fact represents a particular profile of a sub population measured by the *Wage per hour*. The dimensions of the cube are : *Education level* ( $D_1, p_1 = 17$ ), *Professional category* ( $D_2, p_2 = 22$ ), *State of residence* ( $D_3, p_3 = 51$ ), *Household situation* ( $D_4, p_4 = 38$ ), and *Country of birth* ( $D_5, p_5 = 42$ ).

We generate a complete disjunctive table  $Z = [Z_1, Z_2, Z_3, Z_4, Z_5]$  according to a binary coding of the cube dimensions.  $Z$  contains 199 523 rows and  $p = \sum_{t=1}^5 p_t = 170$  columns. By applying the MCA on  $Z$  we obtain  $p - d = 165$  factorial axes  $F_\alpha$ . Each

<sup>1</sup><http://kdd.ics.uci.edu/databases/census-income/census-income.html>

<sup>2</sup><http://kdd.ics.uci.edu/>

**Table 1.** Attribute's test-values of *Professional category* dimension.

$j$	Attributes	Test-values		
		$V_{1j}^1$	$V_{2j}^1$	$V_{3j}^1$
9	Hospital services	-99.90	-99.90	-99.90
14	Other professional services	-99.90	-99.90	99.90
17	Public administration	-99.90	-99.90	99.90
12	Medical except hospital	-99.90	99.90	-99.90
5	Education	-99.90	99.90	99.90
7	Finance insurance	-99.90	99.90	99.90
19	Social services	-99.90	99.90	99.90
8	Forestry and fisheries	-35.43	-8.11	83.57
3	Communications	-34.05	-99.90	99.90
15	Personal services except private	-21.92	-5.50	10.28
13	Mining	-6.59	-99.64	-5.25
16	Private household services	7.77	51.45	11.68
6	Entertainment	40.04	99.90	96.23
1	Agriculture	68.66	3.39	-27.38
4	Construction	99.90	-99.90	-99.90
10	Manufact. durable goods	99.90	-99.90	-99.90
11	Manufact. nondurable goods	99.90	-99.90	-99.90
21	Utilities and sanitary services	99.90	-99.90	-99.90
22	Wholesale trade	99.90	-99.90	-24.37
20	Transportation	99.90	-99.90	99.90
18	Retail trade	99.90	99.90	-99.90
2	Business and repair	99.90	99.90	99.90

axis is associated to an eigenvalue  $\lambda_\alpha$ . Suppose that, according to the histogram of eigenvalues, a user chooses the three first axes ( $k = 3$ ). These axes explain 15.35% of the total inertia of the facts cloud. This contribution does not seem very important at a first sight. But we should note that in a case of a uniform distribution of eigenvalues, we get normally a contribution of  $\frac{1}{p-d} = 0.6\%$  per axis, i.e. the three first axes represent an inertia already 25 times more important than a uniform distribution.

The organized *Census-Income* data cube is obtained by sorting the attributes of its dimensions. For each dimension  $D_t$  its attributes are sorted by the increasing values of  $V_{1j}^t$ , then by  $V_{2j}^t$ , and then by  $V_{3j}^t$ . Table 1 shows the new attributes' order of the *Professional category* dimension ( $D_2$ ). Note that  $j$  is the index of the original alphabetic order of the attributes. This order is replaced by a new one according to the sort of test-values. In the Figures 4 (a) and 4 (b), we can clearly see the visual effect of this arrangement of attributes. These figures display views of data by crossing the *Professional category* dimension on columns ( $D_2$ ) and the *Country of birth* dimension on rows ( $D_5$ ). The representation 4 (a) displays the initial view according to the alphabetic order of attributes, whereas representation 4 (b) displays the same view where attributes are rather sorted according to their test-values.

Remember that the aim of our current approach is not to compress or reduce the dimensions of a data cube. We do not also reduce sparsity of a data representation. Nevertheless, we act on this sparsity and reduce its negative effect on OLAP interpretation. Thus, we arrange differently original facts within a visual effect that gathers them as well as possible in the space representation of the data cube. At a first sight, the visual representation 4 (b) is more suitable to interpretation than 4 (a). We clearly distinguish in Figure 4 (b) four dense regions of full cells. In this regions, the homogeneity is higher than the rest of the space representation of the data cube.

This is confirmed by the measure of homogeneity criterion. Indeed, for a sparsity ratio of 63.42%, the homogeneity criterion for the organized cube of representation 4 (b)

is  $hc(\mathcal{C}_{org}) = 0.17$ ; whereas it measures  $hc(\mathcal{C}_{ini}) = 0.14$  for the initial cube of representation 4 (a), i.e, we release a gain  $g = 17.19\%$  of homogeneity when arranging the attributes of the cube according to test-values.

According to the test of the Equation (4), for each  $t \in \{1, \dots, 5\}$ , we select from  $D_t$  the set of characteristic attributes for the three selected factorial axes. These characteristic attributes give the best semantic interpretation of factorial axes and express strong relationships for their corresponding facts. To avoid great number of possible characteristic attributes per axis, we can consider, for each axis, only the first 50% of attributes having the highest absolute test-values. For instance, in the *Professional category* dimension  $D_2$ , the set  $\Phi_2$  of characteristic attributes correspond to grayed rows in table 1.

In the same way, we apply the test of the Equation (4) on the other dimensions of the cube. In the representation of Figure 4 (b), we clearly see that the zones of facts corresponding to characteristic attributes of the dimensions  $D_2$  and  $D_5$  seem to be more interesting and denser than other regions of the data space representation. These zones contains relevant information and reflect interesting association between facts. For instance, we can easily note that industrial and physical jobs, like construction, agriculture and manufacturing are highly performed by *Native Latin Americans* from Ecuador, Peru, Nicaragua and Mexico for example. At the opposite, *Asians* people from India, Iran, Japan and China are rather concentrated in commerce and trade.

## 7. Experimental Results

We have realized some experiments on the *Census-Income* data cube presented in section 6. The aim of these experiments is to appreciate the efficiency of our approach by measuring the homogeneity gain realized by our MCA-based organization on data representations with different sparsity ratios. To vary sparsity we proceeded by a random sampling on the initial dataset of the 199 523 facts from the considered cube.

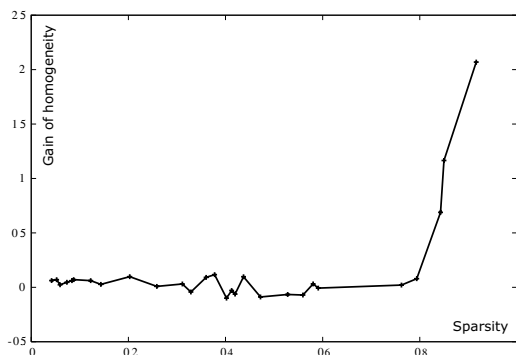
According to Figure 5, the homogeneity gain has an increasing general trend. Nevertheless, we should note that for low sparsity ratios, the curve is rather oscillating around the null value of the homogeneity gain. In fact, when sparsity is less then 60%, the gain does not have a constant variation. It sometimes drops to negative values. This means that our method does not bring a value added to the quality of data representation. For dense data cubes, the employment of our method is not always significant. This is naturally due to the construction of the homogeneity criterion which closely depends on the number of empty and full cells. It can also be due to the structure of the random data samples that can generate data representations already having good qualities and high homogeneity values.

Our MCA-based organization method is rather interesting for data representations with high sparsity. In Figure 5, we clearly see that curve is rapidly increasing to high positive values of gain when sparsity is greater than 60%. Actually, with high relative number of empty cells in a data cube, we have a large manoeuvre margin for concentrating similar full cells and gathering them in the space representation. This shows the vocation of using our approach in order to enhance the visual quality representation, and thus the analysis of huge and sparse data cubes.

	Agriculture	Business and repair services	Communications	Construction	Education	Entertainment	Finance insurance	Forestry and fisheries	Hospital services	Manufact. durable goods	Manufact. nondurable goods	Medical except hospital	Mining	Other professional services	Personal services except private	Private household services	Public administration	Retail trade	Social services	Transportation	Utilities and sanitary services	Wholesale trade
Cambodia																						
Canada		35.0	93.1	54.1			112.5	253.1	182.3			373.4		22.2			169.2	94.0		257.6	11.1	350.0
China	622.0		40.7	50.1			105.0	566.7	336.8	46.7	64.2	60.7					833.8	21.6			329.0	206.3
Columbia							79.0			46.6		80.3					175.0					
Cuba			501.5						31.8			19.0						28.9				
Dominican-Republic			325.0		116.2				148.9	92.7	38.1							35.1	75.0			
Ecuador	107.2	109.1	250.0	205.6	515.0		206.7	68.8			128.1	265.6	100.0				300.0	41.9	175.0		333.3	212.5
El-Salvador	55.6	46.1	36.1	81.0	950.8	344.0			184.7	19.4	120.0			79.5				20.7	400.0	36.9		365.6
England	77.9	222.7	418.1	90.2	50.0	46.9			381.0	257.1	355.0			194.7			136.4	26.3	198.9			
France	450.0										194.8							22.9				
Germany		115.0	200.0	157.1		97.9			417.2	152.3	31.7	128.6		22.2		218.9	108.7	77.9		253.1	428.2	
Greece					257.1				300.0	150.0		241.7					400.0	52.4	400.0			63.6
Guatemala			121.8						47.5	39.8						136.2	25.8					###
Haiti									90.8	80.6						178.7						###
Holand-Netherlands										21.4												
Honduras																151.7		945.0				
Hong Kong	125.4		190.5				590.4	183.3			100.0			225.0	###		150.0			566.7	55.1	484.3
Hungary																400.0						
India		94.2			101.2		17.9	228.1	157.2	145.9								100.0			167.1	181.3
Iran		95.8	225.0				66.7		160.7					311.1	100.0		316.7	90.0		159.0		
Ireland			500.0	100.0						533.3												
Italy					80.3											27.8						
Jamaica	250.0	158.8	###	100.0	147.0		79.2		343.1	571.4	106.0			55.6	91.7	100.0	803.8		604.7	53.3	19.4	
Japan		107.1			63.5	425.0		192.1	678.9	50.9	164.6						26.4	150.0	273.3	107.5		
Laos					500.0				116.6							350.0		71.4				
Mexico	34.5	89.6	75.0	95.0	155.2	46.5	87.6		122.2	61.9	59.8	89.7		159.1	59.9	17.1		52.9	40.3	140.3	121.7	82.1
Nicaragua	159.5		83.3		140.0				47.6	340.0	76.5	65.6		74.1		160.0	178.3	81.0			85.7	
Outlying-U S							###											93.8				200.0
Panama																452.5						
Peru	225.0	699.6	69.7		106.3		47.0	450.0	166.7	215.4	76.2			134.5				127.3	124.2	96.4	20.0	32.0
Philippines	200.0	122.7	245.0	270.0	317.8	62.5			331.1	66.7	146.1	95.6				77.8	134.7		197.3		122.7	
Poland					105.0				325.0	185.5	92.6	175.2					180.0	196.2		187.5		212.5
Portugal					166.7	155.6		107.1		141.1												236.7
Puerto-Rico	87.8	250.0	54.2		66.7	80.7	250.0		37.5	122.3	48.3	420.7		40.0			110.1	23.9	43.5	163.8	142.9	33.6
Scotland					87.5		725.0	300.0		785.0	95.2	14.0		23.9			131.3	350.0	173.8	20.0	36.5	
South Korea																					870.0	
Taiwan																						
Thailand												150.0							43.8			
Trinidad&Tobago	66.3	243.8		63.8	920.0	333.3	89.3		466.7		175.0							453.0	200.0			250.0
United-States	37.8	92.6	153.4	130.6	75.4	117.9	71.1	84.3	214.4	165.4	146.9	141.7		76.0			142.1	99.3	96.0	157.0	199.9	84.4
Vietnam			###				75.0		327.5	173.8				250.0	32.1							
Yugoslavia		42.1																				

(a)

	Hospital services	Other professional services	Public administration	Medical except hospital	Education	Finance insurance	Social services	Forestry and fisheries	Communications	Personal services except private	Mining	Private household services	Entertainment	Agriculture	Construction	Manufact. durable goods	Manufact. nondurable goods	Utilities and sanitary services	Wholesale trade	Transportation	Retail trade	Business and repair services
Philippines	331.1	77.8	95.6	317.8	165.0				265.0				62.5	200.0	270.0	66.7	166.1		322.7	197.3	134.7	122.7
India	157.2			101.2	17.9			228.1					145.9				167.1	81.3		100.0	94.2	
Canada	253.1	22.2	169.2	373.4	54.1			112.5					93.1	182.3				11.1	350.0	267.8	94.0	33.0
Jamaica	343.1	55.6	803.8	106.0	147.0	79.2	604.7		###	91.7		100.0		250.0	100.0		571.4	19.4		533.3		158.8
Iran	311.1	316.7			66.7					100.0										159.0	90.0	95.8
Japan	678.9				63.5		150.0	192.1					425.0			50.9	164.6	107.5		273.3	26.4	107.1
China	336.8		833.8	60.7	50.1	105.0							822.0	40.7	46.7	64.2	329.0	206.3		21.6		
Hong Kong	225.0					590.4	183.3			###			125.4	190.5		100.0	55.1	484.3	366.7		150.0	
Greece	241.7	400.0		257.1	400.0											300.0	150.0		63.6		52.4	
Germany	417.2	22.2	108.7	128.6		97.9			200.0			218.9		157.1	152.3	31.7	428.2		253.1	77.9	115.0	
Scotland	785.0				300.0	350.0					23.9		725.0	87.5	95.2	14.0	700.0	36.5	173.8	134.3		
Poland	325.0		180.0	175.2		105.0			175.6							185.5	92.6		212.5	187.5	196.2	252.9
England	383.0	194.7	136.4		90.2	46.9	198.9		222.7				50.0	418.1	257.1	365.0				26.3	77.9	
Haiti	90.0									178.7						80.6		###				
Taiwan																				46.2		
Panama		452.5																				
Outlying-U S								###											200.0		93.8	
Thailand				150.0																		
Italy				80.3						27.8											32.9	
Hungary												400.0										
Vietnam	327.5	250.0				75.0			###	32.1						173.8						
Holand-Netherlands																	21.4					
Portugal	141.1			155.6	107.1											166.7			236.7			
Yugoslavia																						42.1
South Korea																						
Honduras										151.7												
Cuba	31.8		19.0						501.5							450.0		394.8			28.9	
France																	125.0				229.0	
Cambodia																	92.7	38.1				
Dominican-Republic	146.0				116.7		75.0		375.0												35.1	
Laos		350.0											500.0			116.6					71.4	
Guatemala										136.2		25.8			121.8	47.5	39.8					
Columbia			175.0	80.3		79.0										46.6						
Ireland									500.0							100.0						
Trinidad&Tobago				175.0		333.3	200.0	89.3					920.0	66.3	63.8	466.7			250.0		453.0	243.8
Puerto-Rico	37.5	40.0	110.1	420.7		80.7	43.5	250.0	250.0				86.7		54.2	122.3	48.3	142.9	33.6	163.8	23.9	97.8
Ecuador			300.0	265.6	515.0	206.7	175.0	68.8	250.0		100.0			107.2	205.6		128.1	333.3	212.5		41.9	109.1
Peru	166.7	134.5			106.3	47.0	124.2	450.0	69.7					225.0		215.4	76.2	20.0	32.0	86.4	127.3	699.6
Nicaragua	74.1	178.3	65.6	140.0					47.6	83.3			160.0		159.5	340.0	76.5	85.7			81.0	
Mexico	122.2	159.1		89.7	155.2	67.6	40.3		75.0	59.9		17.1	46.5	34.5	95.0	61.9	59.8	121.7	82.1	140.3	52.9	89.6
El-Salvador				120.0	81.0	344.0	400.0			79.5			900.0	55.6	36.1	184.7	19.4			305.6	38.9	20.7
United-States	214.4	76.0	142.1	141.7	75.4	71.1	86.0	84.3	153.4				116.9	37.8	130.0	165.4	146.9	109.9	84.4	157.0	59.3	92.6



**Figure 5.** Evolution of the homogeneity gain according to sparsity.

## 8. Conclusion and Future Work

In this paper, we introduced a MCA-based approach to enhance the space representation of large and sparse data cubes. This approach aims to provide an assistance to the OLAP user and helps him/her to easily explore huge volumes of data. For a given data cube, we compute the test-values of its attributes. According to these test-values, we arrange attributes of each dimension and so display in an appropriate way the space representation of facts. This representation provides better property for data visualization since it gather full cells expression interesting relationships of data. We also identify relevant regions of facts in this data representation by detecting characteristic attributes of factorial axes. This solve the problem of high dimensionality and sparsity of data and allows the user to directly focus his exploration and data interpretation on these regions. We have also proposed an homogeneity criterion to measure the quality of data representations. This criterion is based on the notion of geometric neighborhood of cells and their measures' similarities. Through experiments we led on real world data, our criterion proved the efficiency of our approach for huge and sparse data cubes.

Currently, we are studying some possible extensions for this work. We consider the problem of optimizing complexity of our approach. We also try to involve our approach in order to take into account the issue of data updates. Finally, we project to implement this approach under a Web environment that offers an interesting on-line aspect and a good user interaction context.

## References

- [1] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, 1996.
- [2] R. Kimball. *The Data Warehouse Toolkit*. John Wiley & Sons, 1996.
- [3] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, 1997.
- [4] R. Ben Messaoud, O. Boussaid, and S. Loudcher Rabaséda. Using a Factorial Approach for Efficient Representation of Relevant OLAP Facts. In *Proceedings of the 7<sup>th</sup> International Baltic Conference on Databases and Information Systems (DB&IS'2006)*, pages 98–105, Vilnius, Lithuania, July 2006. IEEE Communications Society.
- [5] R. Ben Messaoud, O. Boussaid, and S. Loudcher Rabaséda. Efficient Multidimensional Data Representation Based on Multiple Correspondence Analysis. In *Proceedings of the 12<sup>th</sup> ACM SIGKDD Inter-*

- national Conference on Knowledge Discovery and Data Mining (KDD'2006)*, pages 662–667, Philadelphia, PA, USA, August 2006. ACM Press.
- [6] J.P. Benzecri. *Correspondence Analysis Handbook*. Marcel Dekker, hardcover edition, January 1992.
  - [7] L. Lebart, A. Morineau, and M. Piron. *Statistique Exploratoire Multidimensionnelle*. Dunold, Paris, 3<sup>rd</sup> edition, 2000.
  - [8] R. Ben Messaoud, O. Boussaid, and S. Loudcher Rabaséda. Evaluation of a MCA-Based Approach to Organize Data Cubes. In *Proceedings of the 14<sup>th</sup> ACM International Conference on Information and Knowledge Management (CIKM'2005)*, pages 341–342, Bremen, Germany, October – November 2005. ACM Press.
  - [9] J. S. Vitter and M. Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *ACM SIGMOD International Conference on Management of Data (SIGMOD 1999)*, pages 193–204, Philadelphia, Pennsylvania, U.S.A., June 1999. ACM Press.
  - [10] D. Barabási and M. Sullivan. Quasi-Cubes: Exploiting Approximations in Multidimensional Databases. *SIGMOD Record*, 26(3):12–17, 1997.
  - [11] J. Shanmugasundaram, U. M. Fayyad, and P. S. Bradley. Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. In *5<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 1999)*, pages 223–232, San Diego, California, U.S.A., August 1999.
  - [12] Y. Sismanis, A. Deligiannakis, N. Roussopoulos, and Y. Kotidis. Dwarf: Shrinking the PetaCube. In *ACM SIGMOD International Conference on Management of Data (SIGMOD 2002)*, pages 464–475, Madison, Wisconsin, U.S.A., 2002.
  - [13] W. Wang, H. Lu, J. Feng, and J. X. Yu. Condensed Cube: An Effective Approach to Reducing Data Cube Size. In *18<sup>th</sup> IEEE International Conference on Data Engineering (ICDE 2002)*, pages 155–165, San Jose, California, U.S.A., February–March 2002.
  - [14] J. Feng, Q. Fang, and H. Ding. PrefixCube: Prefix-Sharing Condensed Data Cube. In *7<sup>th</sup> ACM International Workshop on Data warehousing and OLAP (DOLAP 2004)*, pages 38–47, Washington D.C., U.S.A., November 2004.
  - [15] L. V.S. Lakshmanan, J. Pei, and J. Han. Quotient Cube: How to Summarize the Semantics of a Data Cube. In *28<sup>th</sup> International Conference of Very Large Data Bases (VLDB 2002)*, Hong Kong, China, August 2002.
  - [16] L. V.S. Lakshmanan, J. Pei, and Y. Zhao. QC-Trees: An Efficient Summary Structure for Semantic OLAP. In ACM Press, editor, *ACM SIGMOD International Conference on Management of Data (SIGMOD 2003)*, pages 64–75, San Diego, California, U.S.A., 2003.
  - [17] Y. Feng, D. Agrawal, A. El Abbadi, and A. Metwally. Range CUBE: Efficient Cube Computation by Exploiting Data Correlation. In *20<sup>th</sup> International Conference on Data Engineering (ICDE 2004)*, pages 658–670, Boston, Massachusetts, U.S.A., March–April 2004.
  - [18] K. A. Ross and D. Srivastava. Fast Computation of Sparse Datacubes. In *23<sup>rd</sup> International Conference on Very Large Data Bases (VLDB 1997)*, pages 116–125, Athens, Greece, August 1997.
  - [19] X. Li, J. Han, and H. Gonzalez. High-Dimensional OLAP: A Minimal Cubing Approach. In *30<sup>th</sup> International Conference on Very Large Data Bases (VLDB 2004)*, pages 528–539, Toronto, Canada, August 2004.
  - [20] Y. W. Choong, D. Laurent, and P. Marcel. Computing Appropriate Representations for Multidimensional Data. *Data & knowledge Engineering Journal*, 45(2):181–203, 2003.
  - [21] R.B. Cattell. The Scree Test for the Number of Factors. *Multivariate Behavioral Research*, 1:245–276, 1966.
  - [22] H.F. Kaiser. A Note on Guttman's Lower Bound for the Number of Common Factors. *Brit. J. Statist. Psychol.*, 14:1–2, 1961.
  - [23] E. Malinvaud. Data Analysis in Applied Socio-Economic Statistics with Special Consideration of Correspondence Analysis. In *Marketing Science Conference*, Jouy en Josas, France, 1987.
  - [24] B. Escofier and B. Leroux. Etude de Trois Problèmes de Stabilité en Analyse Factorielle. *Publication de l'Institut Statistique de l'Université de Paris*, 11:1–48, 1972.

# Model-Driven Development for Enabling the Feedback from Warehouses and OLAP to Operational Systems

Lina NEMURAITĖ<sup>1</sup>, Jurgita TONKUNAITE and Bronius PARADAUSKAS  
*Department of Information Systems, Kaunas University of Technology, Lithuania*

**Abstract.** The goal of this paper is to demonstrate how interfaces of Data Warehouses and OLAP may be used in holistic Model-driven development process of applications of enterprise. Traditionally, the main responsibilities of Data Warehouses and OLAP are concerned with relation to data analyzing and reporting activities. However, the real power of data analysis lies in possibility to support dynamic formulation of queries and analyzable data structures to respond to emerging needs of users and other applications of enterprise and the Web. From the business goals and processes viewpoint, there is no difference between operational systems and Data Warehouses/OLAP. The proposed Model-driven methodology brings possibility to automate development of enterprise applications extended with analytical capabilities, incorporating feedback from OLAP tools into computerized business processes, and tendering analysis results for business improvement. Proposed metamodels and transformations are extension of our previous work, which was focused on generating normalized multidimensional data models on demand. In this paper, wider possibilities to obtaining well-formed warehouse schemas, ensuring completeness of warehouse data, and using them in Model-driven development process are considered.<sup>2</sup>

**Keywords.** Model-driven development, warehouse, OLAP, multidimensional data model, metamodel, transformation, interface

## Introduction

Currently, enterprise applications development has reached the stage, when possibility to manage business solutions on demand is becoming a reality. In [1], a vision and the prototype of software architecture and development process are presented enabling the monitoring of business processes of an enterprise. In this vision, the outputs of Data Warehouse – business performance indicators – are observable on Management Dashboard, where features of business processes may be changed to obtain desirable solutions. Alongside, a Model-driven development (MDD) framework is supported to maintain changes of software components devoted for monitoring and management of changing business processes. Without automated model manipulations, adaptations, validation and transformations to code this vision of an adaptive enterprise would be impossible. For integrating Warehouses and Online Analytical Processing (OLAP) into

---

<sup>1</sup> Corresponding Author: Lina Nemuraite, Department of Information Systems, Kaunas University of Technology, Studentu 50-308, LT-51368, Lithuania; E-mail: lina.nemuraite@ktu.lt.

<sup>2</sup> The work is supported by Lithuanian State Science and Studies Foundation according to Eureka programme project „IT-Europe” (Reg. No 3473).

business management chain, the processes of their development must be incorporated into Model-driven development framework of enterprise applications.

The MDD framework [1] proposed for IBM Business Performance Management Solution represents the mostly matured case per various efforts endeavoring to bridge outputs and inputs of business models and to make feedback from gained business knowledge to business management. Data Warehouses and Online Analytical Processing (OLAP) are playing the important role here. Traditionally, the main responsibilities of Data Warehouses and OLAP were concerned with relation to data analyzing and reporting activities. However, the results of analytical reports may be used directly for adjustment various parameters regulating business activities, for example, calculation of customer discounts or personnel benefits, forecasting of demand etc. Outputs from Business Analytics are still isolated from other applications and cause a gap. A real power of data analysis lies in possibility to support dynamic formulation of queries and analyzable data structures to respond to emerging needs of users and other applications of enterprise and the Web. The current practices of development of Data Warehouses are not suitable for this purpose. Mainly, they suffer from two converse tendencies: long-lasting and heavy-weight development/refreshment processes, when changes “on demand” are hardly imaginable; or “ad-hoc” warehouses, having a little value for business.

For improvement of warehouse development process and making it suitable for automated development, it must be made as simple as possible, and precise. In our previous work [2], a MDD method for warehouse was proposed based on transformations between Primary Concept Model, representing operational data objects, and Warehouse Concept Model, representing multidimensional data objects. The method was devoted for automatic or semi-automatic development of warehouse schema in multi-dimensional normal form ensuring avoiding of aggregation anomalies. In current paper, the method is extended for usage in holistic Model-driven development process of enterprise applications where interfaces of warehouses and OLAP are used evenly with others. The common development process is especially important with the raised analytical capabilities of large Database Management Systems such as Oracle 10g, MS SQL Server 2005, DB2.

Moreover, capabilities of Warehouses and OLAP should be considered in the context of Service-oriented architecture (SOA) that is gaining more and more popularity and becoming the dominant architectural orientation of enterprises. In this paper, our look from positions of Model-driven and Service-oriented development to existing data warehouse and OLAP models is presented. It seems that generic interfaces and data object types are desirable for representation of warehouses and OLAP formulating integrated enterprise service models (but it does not mean that Web services must be used for internal manipulating with warehouse data inside enterprise).

The rest of the paper is organized as follows. In Section 1, the background and related work is concerned. Section 2 presents generic Model-driven development framework including Warehouse and OLAP development process. Section 3 explains the Primary Concept and Warehouse Concept metamodels that give the basis for transformations. In Section 4, an example of using warehouse in Forest Information system is described. Finally, Section 5 draws conclusions and highlights the future work.



## 1. Related Work

Model-driven development, Service-oriented architecture (SOA) and Business Intelligence are today challenges capable to give synergetic effect if they were used together. Developing applications for business process monitoring and managing by implementing knowledge-based feedback from BI tools is a difficult task. Practiced methods for developing such applications are hidden in specific platforms and tools whereas there are needs to include them into common development cycle. For using warehouses and OLAP in generic MDD process together with other (operational) software components of enterprise, the Design-Independent Modeling [3] may be used. Design Independent Model (DIM) was introduced to represent requirements in Model-driven architecture (MDA) related spirit as MDA does not consider requirements model though requirements specifications are inherent for every software project.

In DIM, initial requirements, described by use cases and model of domain under consideration, are mapped to interfaces and entities (data object types) models of targeted system. The method supplements MDA and is intended for service-oriented systems. For introducing Warehouses and OLAP into MDD, we just need to introduce appropriate interfaces and data object types on conceptual level. Such data types already exist in various platform-specific standards as JOLAP [4], Multidimensional Expressions MDX [5], XML for Analysis (XMLA) [6], OLE DB for OLAP (ODBO), OLAP BAPI, Service Data Objects (SDO) [7] and surely others. The most generic data types are SDO, a standard designed to simplify the representation of service-oriented business logic. As Service Component Architecture (SCA) gives developers and architects the ability to represent business logic as reusable components that can be easily integrated into any Service Component Architecture (SCA) compliant application or solution; SDO specify a standard way to access data and can be used to modify business data regardless of how it is physically accessed. SCA and SDO are designed to be used together with the SOA programming model. Using these specifications, organizations can more easily create new and transform existing IT assets into reusable services.

SDO data objects are data graphs (ordered sequences of graphs) together with their metadata. All data object types of warehouse (or multidimensional model) as well as data sources used as inputs of warehouse may be represented as subtypes of data graphs. Using data graphs may considerably simplify warehouse design and its integration into enterprise application development process. Similar ideas are proposed in [8] where primitive, regular and composite object types are used for OLAP and object data model. However, the concept of data graph is more generic, applicable to service-oriented development of all kinds of applications: integration, Extract-Transform-Load (ETL), Data Mining, and others.

There are only a few authors working in Model-driven development of warehouses: already mentioned group [1], Trujillo and Lujan-Mora (for example, [9, 10, 11]), Celma and Zepeda [12]. While Trujillo and Lujan-Mora are concentrated on UML profile for warehouse modeling and transformation to implementation tools (Oracle DBMS), Celma and Zepeda are working on deriving multidimensional schemas from ER metamodel (both methodologies are based on Common Warehouse Metamodel Specification [13]). Other methods are proposed having potential to automate warehouse and OLAP development [14, 15, 16, 17, 18, 19]. In summary, there are three principal approaches proposed for automating development of warehouses: going from object-oriented Platform-independent model (PIM) to PSM

(Trujillo); deriving warehouse schema (PIM) from ER (PIM) model and mapping it to relational PSM (Zepeda); deriving multidimensional model from OLAP requirements (e.g. [14]). There are computation speed and conceptual clarity arguments for using object-oriented multidimensional model against materialized views approach, but it seems likely that there is no one solution to fit every need. However, many unused possibilities exist to apply knowledge from formal methods of multidimensional modeling for automating a hard task of warehouses and OLAP development.

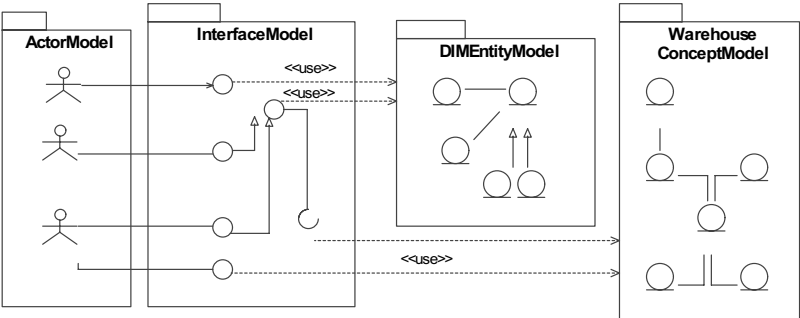
In our work, an attention is given for multidimensional schema design ensuring correctness or normalization that is based on summarizability dependencies between warehouse dimensions and facts. Aggregation conditions and multidimensional normal forms were considered by Lenz [20], Lechtenborger [21,22], Schneider [23], and Mansmann [24]. These works give a background for establishing summarizability constraints that should be captured in warehouse MDD process. For ensuring meaningful queries, aggregation dependencies must be specified between all potential measures and dimensions. The interesting conclusion is that conceptual normalization of operational data model ([25, 26, 27] leads to normalized multidimensional model.

Studying warehouse structures, we came to the issue that the most appropriate schema for warehouse is a snowflake schema, having one or more cubes, shared dimensions, value and level based hierarchies, and even facts related to facts [22] as it suits many practical situations.

Finally, a very important issue is a semantic correctness of warehouse data. Warehouse tools are working with missing values treating them as null values, but it is very important to know the actual database semantics. The conceptual normalization helps to avoid null values in operational database; missing values in warehouse sometimes may be replaced with nulls, forecasted or interpolated values.

**2. Generic Model-Driven Development Framework for OLTP, Warehouses and OLAP Service Components**

In this section, the Design Independent Modeling concepts are applied to warehouses and OLAP that are used evenly with other application components in common Model-driven Service-oriented development models in contrast to current practice of developing warehouses separately, at platform-specific level.



**Figure 1.** Information system requirement model (DIM), including warehouse

In DIM [3], requirements to system under development are obtained from use case model and represented as interfaces consisting of sets of operations specified as events; arguments and preconditions/post-conditions of these operations are expressed in terms of entities (data object types) of problem domain. DIM consists of Actor, Interface and Entity models; for extending DIM to including multidimensional data objects, we must introduce Warehouse Concept Model that was described in [2]. The abstract Design Independent Model capable to represent requirements to Business Analytic applications is presented in Figure 1. Warehouse Concept Model represents requirements to multidimensional data model; in our case (Figure 2), it is derived from operational (DIM Entity) model using intermediate Primary Concept Model described later in Section 3. Transforming requirements to architectural design (PIM) models, data models remain unchanged; interfaces are transformed to services, components or object-oriented design model. In the next phase, operations of interfaces to warehouse and OLAP must be transformed to corresponding operations of services or components of selected tools, or query language (for example, JOLAP and XMLA).

Service Data Objects [7] involve Data Objects, Data Graphs and Metadata (Figure 3). Data Objects represent ordinary and primitive data types, and references to other Data Objects. Data Graphs are conceptual (possibly, multi-rooted) sets of Data Objects; they provide the units of transfer between different service components. Metadata allow getting information about data types, their relationships, and constraints. In summary, SDO provide a common data and metadata interfaces across data source types for different tools and frameworks of enterprise. Consequently, SDO may be specialised for representing multidimensional data types for interfaces to Warehouses, ETL and OLAP tools.

Studying XMLA, we find OLE DB for OLAP data types that may be generalised to the following conceptual data object types (UML 2.0/OCL 2.0 types extended with multidimensional types):

- primitive types (Boolean, Float, Integer) and classes, including Enumeration;
- OCL 2.0 types: Collections, Sets, Sequences;
- multidimensional Types: OLAP, Cube, Axis (Edge), Dimension, Hierarchy, Level, Member, Cell – these types represent MDX concepts [5] in XMLA specification, having equivalents in JOLAP.

### 3. Primary Concept Model and Warehouse Concept Model

Primary Concept Model (Figure 4) represents (simplified) intermediate model between data model described by UML class model with OCL constraints (DIM Entity model [3, 26, 27]), and Warehouse Concept Model. It does not include OCL constraints, describing integrity of conceptual data model, as they are not required for warehouse. Instead, aggregation and derivation constraints are added that are necessary for warehouse concept model.

Primary Concept Model is normalized data model, based on existence dependencies [26]. It may have association and generalization relationships; generalizations are disjoint and complete, so the transformation to Warehouse Concept Model is straightforward. In our previous work, this model was used for implementation of visual ETL tool for rapid development of warehouse schemas with MS SQL Server 2000. Aggregation constraints representing data semantics must be



added manually preparing operational data model for generation of conceptual warehouse models on demand. For semi-automatic development of warehouse, aggregation constraints may be added directly to warehouse model.

Aggregation (or summarizability) constraints define the normalization (or well-formedness) of data warehouse. A data warehouse is a multidimensional database, whose atomic information units are facts. A *fact* is a point in a multidimensional space to which measurable business facts, so-called *measures*, are assigned. Each *dimension* of the multidimensional space is structured as a *hierarchy* of *dimension levels*. A dimension level is an attribute, which is associated with a *domain* of values. Elements of “higher” levels in this hierarchy logically contain elements of “lower” levels, for example, in Time hierarchy the element of higher level “Month” logically contains the elements of lower level “Day”, which allows analyzing multidimensional data at different levels of detail and aggregate facts with respect to dimension levels.

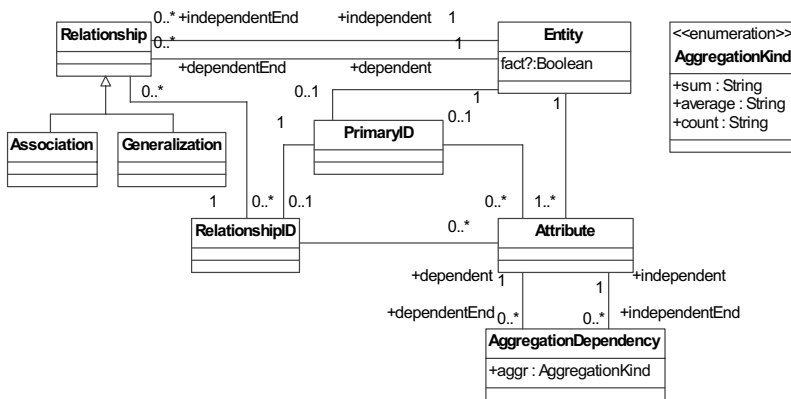


Figure 4. Primary Concept Metamodel

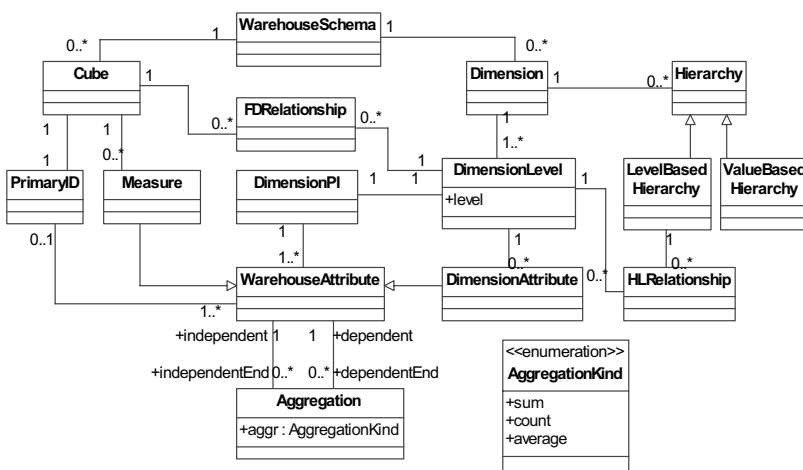


Figure 5. Warehouse Concept Metamodel

Warehouse concepts “Cube”, “Dimension”, “Dimension Level”, “Measure”, “Dimension Attribute”, “Aggregation Dependency”, “Primary Identifier” and others are presented in warehouse metamodel (Figure 5) (simplified variant of OLAP metamodel from [13]). The development of data warehouse involves creating of scheme of data warehouse and data transformations. The scheme of data warehouse depends on queries that are required to executing, while data transformations depend on differences between detail of data persisting in operational database and in warehouse. This stage would become more simple and faster if normalized operational data model is taken as input and normalized multidimensional data model as output. Moreover, it is a faster way to use bulk data loading instead of loading row by row, and using “Extract-Load-Transform” instead of “Extract-Transform-Load” [28] what means copying or replication of tables from operational data source to warehouse and later transformation. Data object type is an abstraction, meaning data table, external source, result of integration etc.

According to our approach, some transformations can be moved to analysis (OLAP) level, some of them must be performed by user while defining data warehouse requirements as these transformations cannot be automated. For example, operational data that are not adequate for analysis – notes, comments, system attributes – should be removed. Bringing derived, different levels of data, creating arrays, and so on, it is purposeful to realize these by OLAP operations. In such a way, the minimal set of data warehouse creation transformations include:

- reorganization of operational structures to structures of warehouse dimensions and facts;
- creation of time dimension that is often needed;
- conversion between different data types;
- adding rating attributes where summary value must be allocated between different hierarchies, or allocating a value in proportion to the number of segregating hierarchies;
- changing generalization relationship to “Type” object type;
- resolving one-to-many associations between facts and dimensions. There are several solutions to this problem, but only two of them are compatible with our principles. One way is to add rating attributes for distributing the summary value between members belonging to one fact; the sum of ratios must equal to one. The second way is to decrease the granularity of a fact (adding ratings as well).

In practice, the rating attributes often are predefined in operational database as in our example of forest information system: the area of parcel is approximately divided between different kinds of trees growing in parcels.

#### **4. The Example of Forest Information System**

The method was tried for Forest Information System (FIS) where historical information about states of forest parcels is stored in MS SQL Server database. Simplified use cases of FIS are presented in Figure 6, interactions between interfaces obtained from these use cases – in Figure 7. Forest checkers are regularly inventorying states of forest parcels every five years. Every year only fifth of the whole terrain is inspected, so information about parcel may be to five years old. During this period, forest enterprises are registering deforesting/reforesting work and accidents changing the forest state. As

trees are growing, for making reasonable forestry plans the forest state must be predicted or interpolated – null values are not allowed instead of missing ones. Planning is performed in a proportion to factors of growing of different kinds of trees and other ratios evaluating various aspects of forest status. Deforesting and reforesting plans are made in top-down manner, incrementally allocating the overall amounts to smaller territory units. Such activities may be efficiently performed using Warehouse, OLAP and Business Intelligence tools. Predictions, interpolations, and forestry planning is performed in warehouse and returned back to operational system.

In Figure 8, the requirement level (substantially simplified) conceptual model of the fragment of FIS is presented. Even for such complex domain, the transformation of Primary Concept Model to Warehouse Concept model is almost straightforward. Rating attributes were used for different kinds of parcel trees (they were transformed to facts of smaller granularity), project parcels (sometimes only part of the parcel is incorporated to the project) and enterprise localities (there are localities belonging

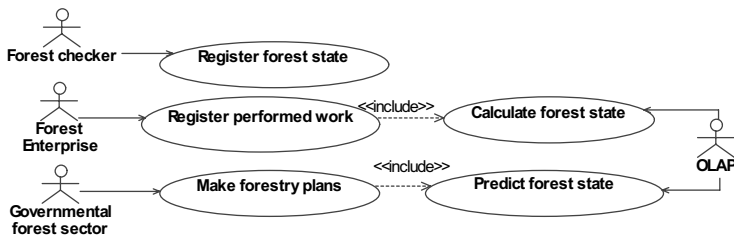


Figure 6. Use cases of Forest Information System

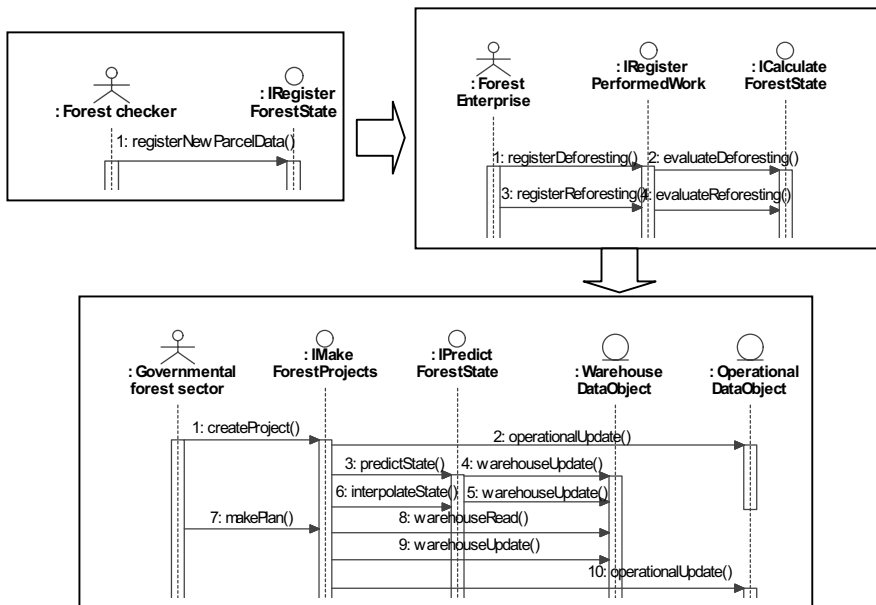


Figure 7. Interactions between interfaces obtained from use cases of Forest Information System





about height and thickness of trees are calculated on the base of previous (or previous and later) values; forecasting parameters are continuously tuned when actual values become available.

## 5. Conclusions and Future Work

In this paper, the challenges of Business Intelligence, Model-driven development and Service-oriented architecture were analyzed. The joint use of such different methodologies is promising as it gives possibility for interoperability and feedback between results, obtained during business analysis, and daily activities of enterprises. Needs for this kind of applications are arising in many problem domains.

In this paper, the methodology is proposed for requirement level modeling of Warehouses and OLAP that are used evenly with other application components in common Model-driven Service-oriented development models in contrast to current practice of developing warehouses separately, at platform-specific level. The methodology is based on representation of requirements initially described by use cases via interfaces and conceptual models of operational data object types (entities) and multidimensional ones (warehouse concepts). Though based on different concepts, both types of data objects may be generalized as Service data objects (SDO) – data graphs; a standard designed to simplify the representation of service-oriented business logic.

A complex and long-lasting Warehouse development process may be significantly improved using Model-driven development. The necessary condition for easier development of warehouses is a normalization of operational and multidimensional data models. Normalized initial data model may be transformed to warehouse model using simple transformations; it leads to normalized multidimensional model and helps to avoid null values in operational database.

Our future work is directed to further elaboration of the joint design methodology and strengthening its support by universal CASE tools, for making the development of meaningful warehouses easy and incorporating them into Service-oriented architecture of enterprise.

## References

- [1] P.Chowdhary et al. Model driven development for business performance management. *IBM Systems Journal* **45** (3) (2006), 587-605.
- [2] J.Tonkunaite, L.Nemuraite, B.Paradauskas. Model-driven development of data warehouses. *Databases and Information Systems, 7th International Baltic Conference, 3-6 July* (2006), 106-113.
- [3] L. Ceponiene, L. Nemuraite. Design independent modeling of information systems using UML and OCL. *Databases and Information Systems*, **118** (2005), Amsterdam, IOS Press, 224-237.
- [4] *Java™ OLAP Interface (JOLAP)*. JSR-69 (JOLAP) Expert Group. Version 1.0. Proposed Final Draft, 30 August 2003.
- [5] G. Spofford, S. Harinath, C. Webb, D. H. Huang, and F. Civardi. *MDX Solutions With Microsoft® SQL Server™ Analysis Services 2005 and Hyperion® Essbase*. Wiley Publishing, Inc., Indianapolis, Indiana, 2006.
- [6] *XML for Analysis Specification*. Version 1.1. Microsoft Corporation, Hyperion Solutions Corporation, 11/20/2002.
- [7] J. Beatty, S. Brodsky, M. Nally, and R. Patel. *Next-Generation Data Programming: Service Data Objects*. BEA Systems, Inc. ("BEA") and International Business Machines Corporation ("IBM"), 2003.

- [8] E. Pourabbas, A. Shoshani, The Composite OLAP-Object data model: removing an unnecessary barrier. *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, 2006 291-300.
- [9] J. Trujillo, M. Palomar, J. Gómez, I-Y. Song. Designing data warehouses with OO conceptual models. *IEEE Computer* **34** (12) (2001), 66-75.
- [10] S. Luján-Mora, J. Trujillo, and I.Y. Song. A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering* (to appear).
- [11] R. Villarroel, E. F. Medina, M. Piattini, J. Trujillo, A UML 2.0/OCL extension for designing secure data warehouses. *Journal of Research and Practice in Information Technology* **38** (1) (2006), 31-43.
- [12] M. Celma, L. Zepeda, A model driven approach for data warehouse conceptual design, *Databases and Information Systems, 7th International Baltic Conference, 3-6 July 2006*, 114-121.
- [13] *Common Warehouse Metamodel (CWM) Specification*. OMG document pas/06-04-02, Version 1.1, ISO/IEC 19504:2006(E).
- [14] A. Nabli, J. Feki, F. Gargouri, Automatic construction of multidimensional schemas from OLAP requirements, *Computer Systems and Applications. The 3rd ACS/ISSS International Conference*, 2005, 21-28.
- [15] D. Dori, R. Feldman, and A. Sturm, An OPM-based method for transformation of operational system model to data warehouse model, *Software - Science, Technology and Engineering, 2005. Proceedings of IEEE International Conference on 22-23 Feb. 2005*, 57 – 66.
- [16] V. Peralta, A. Illarze, R. Ruggia, Towards the automation of data warehouse logical design: a rule-based approach, *CAiSE'03 Forum, The 15th Conference on Advanced Information Systems Engineering*, Velden, Austria, 16 - 20 June, 2003.
- [17] C. Phipps, K. Davis, Automating data warehouse conceptual schema design and evaluation, *Proceedings of the Intl. Workshop on DMDW'02*, Canada, 2002.
- [18] N. Tryfona, F. Busborg, and J. G. B. Christiansen, starER: a conceptual model for data warehouse design. *International Workshop on Data Warehousing and OLAP*, 1999, 3-8.
- [19] C. Phipps, K. Davis, Automating data warehouse conceptual schema design and evaluation, *Proceedings of the Intl. Workshop on DMDW'02*. Canada, 2002.
- [20] H. J. Lenz, A. Shoshani, Summarizability in OLAP and Statistical Data Bases. *Scientific and Statistical Database Management, 1997. Proceedings Ninth International Conference, 11-13 Aug 1997*, 132-143.
- [21] B.Hüsemann, J. Lechtenbörgel, G. Vossen, Conceptual data warehouse design. *Proceedings of the Intl. Workshop on DMDW'00*, Sweden, 2000.
- [22] J. Lechtenborger, G. Vossen, Multidimensional normal forms for data warehouse design, *Information Systems*, **28** (5) (2003), 415-434.
- [23] M.Schneider, Well-formed data warehouse structures. *Design and Management of Data Warehouses 2003, Proceedings of the 5th Intl. Workshop DMDW'2003*, Berlin, Germany, September 8 2003, 77.
- [24] S. Mansmann, M. H. Scholl, Extending visual OLAP for handling irregular dimensional hierarchies, *Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'06)*. Krakow, Poland, Springer Verlag, 2006, 95–105.
- [25] B. Paradauskas, L. Nemuraite, *Data Bases and Semantic Models*. Monograph. Kauno technologijos universitetas, Kaunas, Technologija, 2002.
- [26] L. Nemuraite, B. Paradauskas. From use cases to well structured conceptual schemas. *Information Systems Development. Advances in Theory, Practice, and Education*. Springer, 2005, 303-314.
- [27] E. Miliauskaitė, L. Nemuraite. Representation of integrity constraints in conceptual models. *Information Technology and Control* **34** (4) (2005), 307-317.
- [28] Is ETL Becoming Obsolete? - a Sunopsis. White Paper. Sunopsis, 2005.

# Data Quality and Data Analysis

This page intentionally left blank

# A Time-Series Representation for Temporal Web Mining Using a Data Band Approach

Mireille SAMIA<sup>1,2</sup> and Stefan CONRAD

*Institute of Computer Science,  
Databases and Information Systems,  
Heinrich-Heine-University Düsseldorf, Germany*

**Abstract.** The analysis of events ordered over time and the discovery of significant hidden relationships from this temporal data is becoming the concern of the information society. Using temporal data as temporal sequences without any pre-processing fails to find key features of these data. Therefore, before applying mining techniques, an appropriate representation of temporal sequences is needed. Our representation of time series can be used in different fields, such as aviation science and earth science, and can also be applied to, for instance, Temporal Web Mining (TWM) [1], [2], [3], [4]. Our representation of time series aims at improving the possibility of specifying and finding an important occurrence. In our new concept, we use data band ranges and areas in order to determine the importance or the weight of a segment. According to the closeness of a segment to a data band range, this representation of time series can help to find a significant event. This paper focuses on our representation of time series.

**Keywords.** Temporal Web Mining, time series, data representation

## Introduction

Uncovering significant hidden information from temporal data is highly needed for the advance of the information society. Data with temporal information are constantly generated and gathered in different areas, such as aviation science and earth sciences. Using and analyzing temporal data as an ordered collection of events contribute to the progress of such domains. However, using temporal data as temporal sequences without any pre-processing fails to retrieve relevant features of these data. Therefore, before applying mining techniques, an appropriate representation of temporal sequences is needed.

Our representation of time series consists of data band ranges and segments. The closeness of a segment to a data band range can be a clue to find an important event. Our representation of time series can be applied to, but is not limited to, Temporal Web Mining [1], [2], [3].

---

<sup>1</sup>This work is supported by the German Academic Exchange Service (Deutscher Akademischer Austauschdienst DAAD).

<sup>2</sup>Corresponding Author: E-mail: mireillesamia@hotmail.com

Temporal Web Mining (TWM) extends temporal data mining and Web mining [1], [2], [3]. It mainly aims at mining temporal data in real time over the Web. Temporal Web Mining uses Web data with temporal information in the temporal data mining process, and intends to introduce prediction as a main issue in Web mining [3].

TWM can be applied to different domains, such as aviation science and earth science. One of its application scenarios is volcanic ash and aircrafts [5]. Volcanic ash consists of tiny jagged particles of rocks and natural glass blasted into the air by a volcano [6]. When wet, volcanic ash conducts electricity, and can present significant hazard to aircraft safety by, for instance, interrupting the power generation and damaging electronics. Combining and analyzing historical volcanic activities, historical incidents of aircrafts entering ash clouds, meteorological data, volcano data and detecting volcanic ash cloud dispersal, in real time and over the Web, can help to prevent an air crash or a danger to an aircraft from volcanic hazards, namely volcanic ash. Volcanic ash can be carried thousands of miles by wind [6]. Considering wind speed is important in the analysis of the dispersal of volcanic ash. Using our representation of time series in the Temporal Web Mining process helps to extract relevant features of the temporal data, namely the wind speed data.

In our time series representation, we define two main types of data band ranges, which accordingly can make possible to discover any early sign of an important event. We denote the first data band range the *Dangerous Data Band* (DDB), and the second data band range the *Risky Data Band* (RDB). DDB consists of the data during the occurrence of a significant event, including the values of high wind speed. RDB consists of the data before the occurrence of a crucial event, such as a volcanic ash dispersal by wind. By estimating the closeness of a segment to RDB or to DDB, the capability of specifying a crucial occurrence, such as a volcanic ash dispersal by wind, is enhanced. In other words, the closeness of a segment to a data band range can be a clue to discover a significant event.

Our representation of time series facilitates the exploration of data. To every segment, we assign a type according to the data band range it belongs to. This helps to recognize the importance of a segment of a subsequence. For instance, if a segment belongs to DDB, then, its importance is higher than the importance of a segment which is in RDB.

In this paper, section 1 provides a brief overview of Temporal Web Mining, and discusses related work. In section 2, we emphasize on our time series representation [2], [3], [4], [5]. In section 3, we apply an instance of wind speed time series to our representation of time series. Section 4 concludes this paper, and points out future direction.

## 1. Overview and Related Work

This section provides a brief overview of Temporal Web Mining (TWM), and discusses further related work.

### 1.1. Overview of Temporal Web Mining

Temporal Web Mining (TWM) concerns the Web mining of data with temporal information [1], [2], [3].

We define Temporal Web Mining as the process of discovering, extracting, analyzing, and predicting data with significant temporal information from the temporal information discovered by the application of temporal data mining techniques, and applying Web mining techniques to this data in real time over the Web [1].

Temporal data uncovered by the application of temporal data mining techniques are used in the Web mining process in order to extract useful data with temporal information in real time over the Web. The derived useful data with temporal information discovered by the application of Web mining techniques are used again in the temporal data mining process [2], [3].

Temporal Web Mining aims at predicting consequences of actions and at associating a cause with a consequence, whenever possible, in order to give potentially useful information about future events. This can lead to the progress of different domains, including environmental science, earth science and aviation science. TWM combines and extends temporal data mining and Web mining in order to deal with real-time data and multiple sequences. As described in [3], real-time data received by one or multiple sources are analyzed using temporal data mining techniques in order to uncover new temporal information. The new temporal information is sent over the Internet and is combined with auxiliary data from other sources. Then, it is used in the Web mining process in order to discover new potential temporal data in real time and to predict previously unknown temporal information.

In the TWM process, the purpose of the analysis of data is to gain understanding of some events. In order to understand the data to be analyzed, such as time-series data, a representation of these temporal data is required. Our representation of time series supports and is applied to Temporal Web Mining, since it aims at finding, interpreting, and discovering to a certain extent a significant event.

### *1.2. Further Related Work*

The presentation and type of information play a key role in the representation of data [7]. The discovery and analysis of relations between data with temporal information starts with the representation of the temporal data. Data are represented into an appropriate form before defining the similarity measures between sequences, and applying mining techniques [8], [9].

An instance of data to be analyzed is time-series data. Since a direct manipulation of continuous real-valued data in an efficient way is difficult, representing data into time series facilitates the extraction of their key features which can support their further analysis. The data are represented into time series by either keeping it in its original form ordered by their instant of occurrence and without any pre-processing [10], or by finding a piecewise linear function able to approximately describe the entire initial sequence. In the method of finding piecewise linear approximations, the original sequence is divided into several segments which are represented by their own linear function. This type of data representation detects the points where a significant change in the sequence happens.

Following [8] and [11], another concept to represent time-series data is segmenting a sequence by iteratively merging two similar segments. Reference [11] presents a method for pattern matching using linear segments, and describes an approach for segmenting time series. In this approach, all segments of a time series have equal importance. An extension to this method, found in [12] and [13], where piecewise linear segmentation is

used to represent a time series, is to associate a weight value with each segment in order to define the importance of each segment according to the entire sequence. Assigning weights to different parts of a time series is important, since it facilitates the comparison of sequences by looking at their most important segments.

Our data representation deals with time series, and can be applied to other than Temporal Web Mining. It can be used in different fields, such as medicine, environmental science, earth science, and finance. In medicine, Electroencephalography (EEG) data can be divided into different frequency ranges, which can denote different brain rhythms. Every brain rhythm represents a specific brain activity, such as eye opening and deep sleep. Every data band range can denote a frequency range. This makes possible to organize and to represent the data into different groups or ranges having different importances. Our representation of time series can also be applied to earth science. Our representation of time series can help to show different measurements of seismic energy at different periods, such as vibration periods. Another application of our representation of time series is in finance. It can help to represent stock data. For example, the data band ranges can depict stock price ranges. Our time series representation helps to spot the key events of a stock's price movement over a period of time. The advantage of our time series representation is that it can group data, and at the same time, represent these data which often denote a certain task or event. The use of the data band ranges divides the data into different ranges or groups having different importance.

The new concept behind our representation is the use of data band ranges and the assignment of a type to each segment. A type is assigned to each segment according to the data band range it belongs to, and can be determined without the need of calculation by exploring to which data band range it belongs to.

Our representation of time series intends at inserting the idea of finding an important event, while representing data. The closeness of a segment to a data band range and the type of a segment can be a clue to find a crucial event. In other words, the new concept behind this representation is to offer the possibility to discover, interpret, and find to a certain extent, a significant occurrence by exploring and by examining the type of a segment using the corresponding graph and data table, and by evaluating the closeness of a segment to a data band range [4]. The idea of adopting data band ranges and areas to determine the type and importance of a segment is new. We define two main types of data band ranges [4], [14]; the *Dangerous Data Band* (DDB) and the *Risky Data Band* (RDB). Afterwards, we subdivide a sequence to subsequences. Each subsequence is divided into equidistant subintervals in order to represent it by a sequence of straight-line segments. Every segment is assigned a type and a weight. According to DDB and RDB, the type of a segment is specified. The weight of a segment is computed based on the area under a subsequence. To estimate the closeness of a segment to a data band range, we evaluate the area that separates it from the *Dangerous Data Band* or from the *Risky Data Band* and compare it to the area under a subsequence. The closer is a segment to the *Dangerous Data Band*, the greater is its weight. Furthermore, estimating the closeness of a segment to DDB or to RDB helps to find an early warning of a crucial event.

## 2. Our Representation of Time-Series Data

This section provides our representation of time-series data.

Our representation of time series [2], [4], [5], [14] mainly consists of segments and



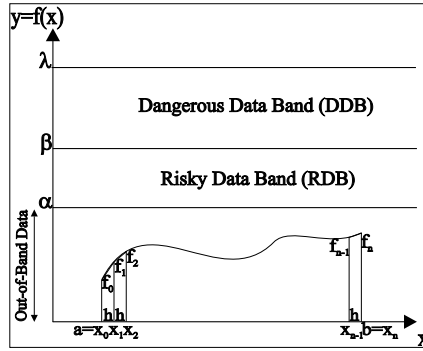


Figure 1. General representation of a subsequence.

data band ranges. To every segment, we assign a type according to the data band range it belongs to. We use data band ranges and areas in order to determine the importance or weight of a segment. The closeness of a segment to a data band range and the type of a segment can be a clue to find an early clue of an important event (such as the dispersal of volcanic ash by wind).

We define two main types of data band ranges [14]. In Figure 1, our first data band range is known as the *Dangerous Data Band* (DDB), and consists of the data during the occurrence of an important event, such as the values of the wind speed. Our second data band range, denoted as the *Risky Data Band* (RDB), consists of the data before an important occurrence (such as high wind speed) happens. RDB is normally close to DDB. Out-Of-Band Data consists of the data outside RDB and DDB. According to each event, more than one RDB, DDB, or Out-Of-Band data range can be defined. For example, based on the Beaufort wind speed scale [15], we can specify the data band ranges, such as RDB and DDB. RDB can be greater or equal to 25 knots and strictly less than 34 knots. DDB can vary between 34 and 47 knots. Analyzing and forecasting the wind speed is strongly required to understand the dispersal of volcanic ash. After specifying the data band ranges, we subdivide the wind speed sequence into subsequences [2], [4], [5], [14]. In Figure 1, we divide each subsequence into equidistant subintervals of length  $h$  in order to represent it by a chain of straight-line segments. We assign to every segment a type and a weight. According to the data band range, we determine the type of a segment. Based on the area under a subsequence, we calculate the weight of a segment. We estimate the closeness of a segment to a data band range by computing the area that separates it from DDB or from RDB, and comparing it to the area under a subsequence. The closeness of a segment to the *Dangerous Data Band* and the *Risky Data Band* helps to specify or find any early clue of a crucial occurrence.

### 2.1. Specifying the Out-Of-Band, the Dangerous Data Band, and the Risky Data Band

In Figure 1, our *Dangerous Data Band* (DDB) varies between  $\beta$  and  $\lambda$ , and our *Risky Data Band* (RDB) between  $\alpha$  and  $\beta$  (i.e.,  $\beta \leq \text{DDB} \leq \lambda$ , where  $\beta < \lambda$  and  $\alpha \leq \text{RDB} < \beta$ , where  $\alpha < \beta$ ).

According to DDB and RDB, we can assign a type to a segment. In other words, we can specify if a data point  $(x_i, y_i)$  is a dangerous point, a risky point or an out-of-band point (i.e., a normal point). More clearly,

- if  $(x_i, y_i)$  is a dangerous point, then  $\beta \leq y_i \leq \lambda$ ,  
where  $\beta < \lambda$  and  $0 \leq i \leq n$ .
- if  $(x_i, y_i)$  is a risky point, then  $\alpha \leq y_i < \beta$ ,  
where  $\alpha < \beta$  and  $0 \leq i \leq n$ .
- if  $(x_i, y_i)$  is an out-of-band point, then  $y_i < \alpha$ ,  
where  $0 \leq i \leq n$ .

The data band ranges can be defined by the user. More than one *Dangerous Data Band*, *Risky Data Band*, or *Out-Of-Band* data range can be specified. Consequently, if  $y_i$  is greater than  $\lambda$ , then  $(x_i, y_i)$  can be a dangerous point, a risky point, or an out-of-band point.

Specifying the type of a segment facilitates data interpretation and visual data analysis. By examining the corresponding graph, we can determine the type of segment based on the data band range it belongs to. Furthermore, according to the closeness of the segment to a data band range, we can speculate its importance or weight.

## 2.2. Segmenting a Subsequence

Sequences are divided into significant subsequences at meaningful breaking points using differentiation. A breaking point is a point in a sequence which corresponds to a starting point of a subsequence or/and to an endpoint of a subsequence.

In our representation of time series, we represent a subsequence by a chain of straight-line segments. We subdivide the subsequence into  $n$  equidistant subintervals of length  $h$ . More specifically, in Figure 1, we subdivide the interval  $[a, b]$  into  $n$  subintervals of length  $h$ . We evaluate the segment joining the data points  $(x_i, f(x_i))$  and  $(x_{i+1}, f(x_{i+1}))$  of the subsequence using the following equation

$$y = f_i + \frac{1}{h}(f_{i+1} - f_i)(x - x_i), \quad (1)$$

where  $f(x_i) = f_i$  and  $f(x_{i+1}) = f_{i+1}$ .

A weight value  $w_i$  is assigned to every segment. After all the subsequences of a sequence are segmented, all the weights are initialized to 1. Therefore, if any of the weights are changed, the weights are renormalized [13]. In order to estimate the weight value of a segment, we evaluate the area of each subinterval  $A_i$ .

In the next sections, we compute the area of each subinterval  $A_i$  of a subsequence. Then, we use  $A_i$  to evaluate the weight of a segment.

## 2.3. Approximating the Area under a Subsequence

In Figure 2, subdividing the subsequence into  $n$  subintervals of length  $h$  and then finding the straight-line segment of each subinterval leads to  $n$  trapezoids. Hence, the area  $A_i$  under each segment of a subsequence is equal to the area of the trapezoid  $T_{f_{i-1} f_i x_i x_{i-1}}$  (cf. Figure 2).

We calculate the area  $A$  of the whole subsequence as follows

$$A = \sum_{i=1}^n A_i, \quad (2)$$

where  $A_i$  is the area under a segment.

The area  $A$  under a subsequence is the summation of the area  $A_i$  of each trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  (cf. Figure 2). To approximate the area  $A$  under a subsequence without computing the area  $A_i$  of each trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  and adding them, we can use the trapezoidal rule. We calculate the area of the  $n$  trapezoids  $T_{f_{i-1}f_i x_i x_{i-1}}$  as follows

$$A = \frac{h}{2}(f_0 + f_n + 2 \sum_{i=1}^{n-1} f_i).$$

Thus, the area under a subsequence is estimated using the trapezoidal rule in the form

$$A = \frac{h}{2}(f_0 + f_n) + h \sum_{i=1}^{n-1} f_i. \quad (3)$$

In the next sections, we estimate the weight value  $w_i$  of each segment. Then, we determine the closeness of the corresponding segment to a data band range by comparing the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  and the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$  (cf. Figure 2).

#### 2.4. Evaluating the Weight of a Segment

We calculate the weight of a segment as a function of the area  $A_i$ , as follows

$$w_i = \frac{A_i}{nA}, \quad (4)$$

where  $A_i$  is the area under a segment,  
 $A$  is the area of the whole subsequence,  
and  $n$  is the total number of segments.

We use the total number of segments  $n$  in order to renormalize the weights. Whenever a segment is inserted, all the weights are changed. More clearly, if one of the weights is changed, all the weights are redistributed.

In the next section, we determine the closeness of a segment to a data band range by comparing the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  and the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$  (cf. Figure 2).

#### 2.5. Estimating the Closeness of a Segment to a Data Band Range

To estimate the closeness of a segment to a data band range, we compare the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  and the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$  (where  $i$  varies between 1 and  $n$ ) (cf. Figure 2). For instance, in Figure 2, the greater is the area of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  according to the area of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$ , the closer is the segment  $f_{i-1}f_i$  to the *Dangerous Data Band* (DDB) or to the *Risky Data*

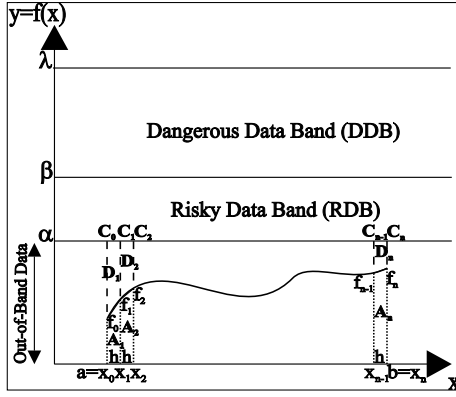


Figure 2. The segments are below RDB and DDB.

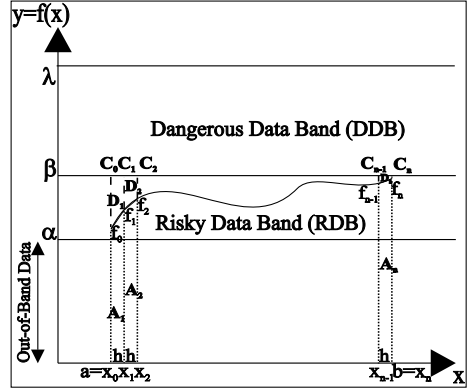


Figure 3. The segments belong to RDB.

Band (RDB). In other words, the greater is the weight of this segment according to the weight of the whole subsequence.

In the previous sections, we found the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  and the weight of a segment  $f_{i-1}f_i$ . To compute the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$ , we calculate the area of the rectangle  $R_{x_{i-1}x_i C_i C_{i-1}}$ . Then, we subtract the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  from the area of the rectangle  $R_{x_{i-1}x_i C_i C_{i-1}}$ . We get the following [14]

$$D_i = \left| h\psi - \frac{h}{2}(f_i + f_{i-1}) \right| = \left| h\left(\psi - \frac{1}{2}(f_i + f_{i-1})\right) \right|, \quad (5)$$

where  $\psi$  is the height of the rectangle  $R_{x_{i-1}x_i C_i C_{i-1}}$  and  $1 \leq i \leq n$ .

From the value of the maximum of  $f_{i-1}$  and  $f_i$ , we determine the value of the height  $\psi$  of the rectangle  $R_{x_{i-1}x_i C_i C_{i-1}}$ . The value of  $\psi$  can be equal to our risky point  $\alpha$ , our dangerous point  $\beta$  or our dangerous point  $\lambda$ .

As previously stated, the closeness of a segment to a data band range is estimated by computing and comparing the area  $A_i$  and the area  $D_i$ . According to the following cases, we can determine the closeness of a segment to a data band range:

*Case 1* (cf. Figure 2) [2], [5], [14]

If  $\max(f_{i-1}, f_i) < \alpha$ , then  $\psi = \alpha$ , where  $\alpha$  is the value of our risky point.

In this case, the smaller is the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  according to the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$ , the greater is the distance of the segment  $f_{i-1}f_i$  to RDB. In other words, the lower is the weight of this segment according to the weight of the whole subsequence.

In this case, the weights of the segments represent the lowest weights of the whole subsequence, because they do neither belong to the *Risky Data Band* nor to the *Dangerous Data Band*.

*Case 2* (cf. Figure 3) [2], [5], [14]

If  $\alpha \leq \max(f_{i-1}, f_i) < \beta$ , then  $\psi = \beta$ , where  $\alpha$  is the value of our risky point, and  $\beta$  is the value of our dangerous point.

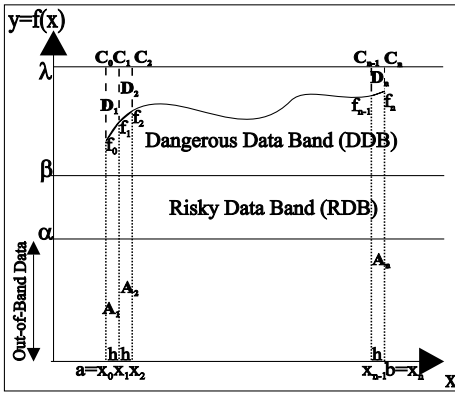


Figure 4. The segments belong to DDB.

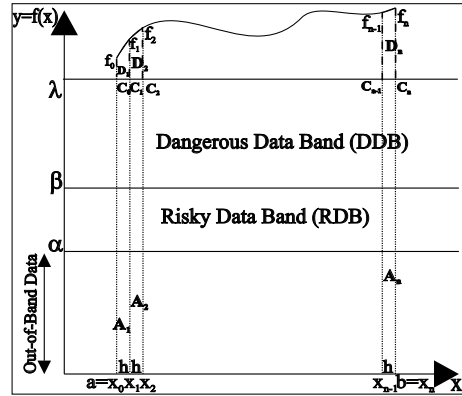


Figure 5. The segments are above RDB and DDB.

In this case, the greater is the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  according to the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$ , the closer is the segment  $f_{i-1}f_i$  to DDB. In other words, the greater is the weight of this segment according to the weight of the whole subsequence.

*Case 3* (cf. Figure 4) [2], [5], [14]

If  $\beta \leq \max(f_{i-1}, f_i) \leq \lambda$ , then  $\psi = \lambda$ ,

where  $\beta$  and  $\lambda$  are the values of our dangerous points.

In this case, the greater is the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  according to the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$ , the closer is the segment  $f_{i-1}f_i$  to the limit of DDB. In other words, the greater is the weight of this segment according to the weight of the whole subsequence. The weights of the segments in this DDB represent the highest weights of the whole subsequence, because they belong to the *Dangerous Data Band*.

In Case 1, Case 2, and Case 3, the area  $D_i$  is a decreasing function of the area  $A_i$ , and the weight is an increasing function of the area  $A_i$ .

*Case 4* (cf. Figure 5) [2], [4]

If  $\max(f_{i-1}, f_i) > \lambda$ , then  $\psi = \lambda$ ,

where  $\lambda$  is the value of our dangerous point.

In this case, we can specify more than one *Dangerous Data Band* (DDB), *Risky Data Band* (RDB), or *Out-Of-Band* data range.

In Figure 5, if we specify an *Out-Of-Band* data range or a *Risky Data Band* (RDB) above DDB, then the greater is the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  according to the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$ , the greater is the distance of the segment  $f_{i-1}f_i$  to DDB. In other words, the lower is the weight of this segment according to the weight of the whole subsequence.

Hence, the area  $D_i$  is an increasing function of the area  $A_i$ , and the weight is an decreasing function of the area  $A_i$ .

In Figure 5, if we specify another *Dangerous Data Band* above DDB, then the greater is the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  according to the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_i C_i C_{i-1}}$ , the greater is the distance of the segment  $f_{i-1}f_i$  to DDB. In other words, the greater is the weight of this segment according to the weight of the whole subsequence.

Hence, the area  $D_i$  is an increasing function of the area  $A_i$ , and the weight is an increasing function of the area  $A_i$ .

By defining a data band, such as the *Dangerous Data Band* (DDB) or the *Risky Data Band* (RDB), we assign for each segment a type, which is determined without the need of calculation and according to the data band range it belongs to. The closeness of a segment to a data band range and the type of a segment can be an early sign of a crucial event. The higher is the weight of a segment according to the weight of the whole sequence, the more important is the event (such as the high wind speed) which is represented by the segment.

### 3. Our Representation of a Wind Speed Time Series

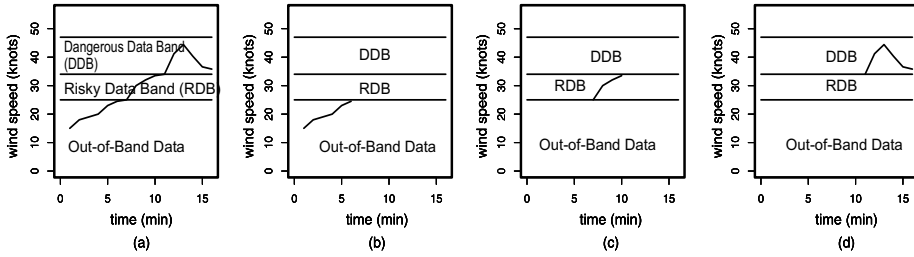
This section applies an instance of a wind speed sequence to our representation of time series.

Time-series data appears in different domains, namely weather forecasting and aircraft operations. Wind-speed data is an example of time-series data. Wind speed is the rate, in knots, kilometers per hour or miles per hour, at which the wind passes a given point [16]. High wind speed can cause dramatic events [2], [5]. For instance, once a volcanic eruption occurs, volcanic ash can be carried thousands of kilometers by wind. Wind speed can affect the volcanic ash fall on the ground and also the dispersal of volcanic ash in the air. An aircraft can enter a volcanic ash cloud without knowing it, and damage all aircraft's engines in less than a minute. Hence, estimating a volcanic cloud dispersal is crucial to aviation safety.

Volcanic ash dispersal is affected by the wind speed. The wind speed can be estimated by using the Beaufort wind speed scale [15], [16]. In Figure 6, wind speed is given in knots [5]. One knot is equal to 1.9 kilometers per hour. According to the Beaufort wind speed scale, we specify our data band ranges. Figure 6(a) denotes a wind speed sequence, which is divided into 3 subsequences belonging to *Out-Of-Band Data* range (cf. Figure 6(b)), to RDB (cf. Figure 6(c)) and to (cf. Figure 6(d)). Note that, for clarity, each subsequence is presented in a separate figure. In Figure 7, the *Out-of-Band* data range is greater or equal to 0 and strictly less than 25 knots. In Figure 8, the *Risky Data Band* (RDB) is greater or equal to 25 knots and strictly less than 34 knots. In Figure 9, the *Dangerous Data Band* (DDB) is between 34 and 47 knots, and is classified as gale in the Beaufort wind speed scale. Using the Beaufort wind speed scale, the values greater than 47 knots can also be divided to more than one dangerous data band. However, in our example, we consider only one out-of-band data, one risky data band and one dangerous data band.

The type of segment of a subsequence is specified according to the data band range it belongs to [2], [4]. If a segment is in the *Out-Of-Band* data range, then it is an out-of-band segment (i.e. normal segment). If a segment belongs to the *Risky Data Band* (RDB), then it is a risky segment. If a segment is in the *Dangerous Data Band* (DDB), then it is a dangerous segment.

Specifying the type of a segment facilitates visual data analysis. By examining to which data band range a segment belongs, the importance or weight of a segment can be speculated. For instance, if a segment belongs to RDB (i.e., a risky segment), then



**Figure 6.** (a) An example of a wind speed sequence. (b) The subsequence is below the Risky Data Band (RDB) and the Dangerous Data Band (DDB). (c) The subsequence belongs to the Risky Data Band (RDB). (d) The subsequence belongs to the Dangerous Data Band (DDB).

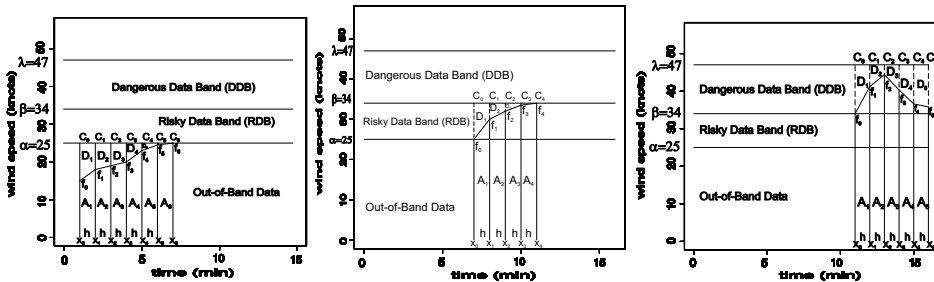
its importance or its weight is lower than that which belongs to DDB (i.e. a dangerous segment).

We subdivide the wind speed sequence of Figure 6(a) into 3 subsequences [2], [4]: The first subsequence is below RDB and DDB and belongs to the Out-Of-Band data range (cf. Figure 7). The second subsequence belongs to RDB (cf. Figure 8), and the third subsequence belongs to DDB (cf. Figure 9). Note that for clarity, each subsequence is illustrated in a separate figure. More specifically, we subdivide every subsequence into equidistant subintervals of length  $h$ , where  $h$  is equal to 1 (cf. Figures 7, 8, 9). Every subsequence is a piecewise-linear function, and is defined on a sequence of intervals. In Figure 7, the subsequence belonging to the Out-Of-Band data range is defined on the interval  $[1, 7]$ , and is composed of 6 segments. In Figure 8, the subsequence belonging to RDB is defined on  $[7, 11]$ , and has 4 segments. In Figure 9, the subsequence belonging to DDB is defined on the interval  $[11, 16]$ , and consists of 5 segments.

To estimate the weight of a segment, we compute the area  $A_i$  under each segment of a subsequence; i.e. the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  (cf. Figures 7, 8, 9) by applying the Equation (3) of the previous section. The values of the area  $A_i$  are presented in Table 1.

Having found the area  $A_i$ , we calculate the weight of a sequence as a function of the area  $A_i$  by applying Equation (4). In our example, the weight is an increasing function of the area  $A_i$  (cf. Figure 10).

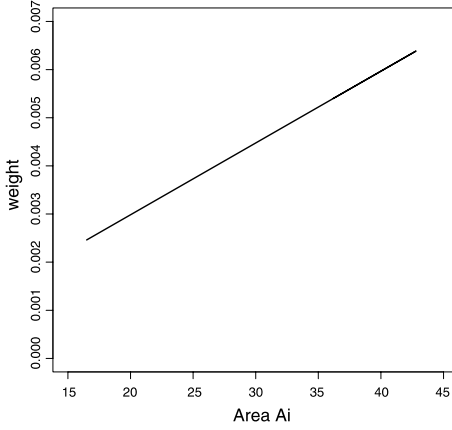
To evaluate the closeness of a segment to a data band range, namely to RDB or to DDB, we compare the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_i x_i x_{i-1}}$  and the area  $D_i$  of the



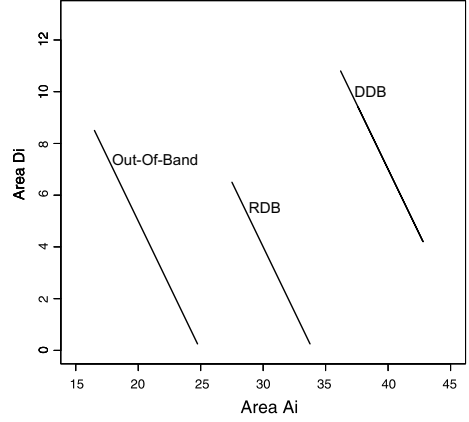
**Figure 7.** The segments are below RDB and DDB.

**Figure 8.** The segments belong to RDB.

**Figure 9.** The segments belong to DDB.



**Figure 10.** The weight of a sequence is an increasing function of the area  $A_i$ .



**Figure 11.** The area  $D_i$  belonging to Out-Of-Band, RDB, or DDB is a decreasing function of the area  $A_i$ .

trapezoid  $T_{f_{i-1}f_iC_iC_{i-1}}$ . Using Equation (5), we compute the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_iC_iC_{i-1}}$ ; i.e., we subtract the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_ix_ix_{i-1}}$  from the area of the rectangle  $R_{x_{i-1}x_iC_iC_{i-1}}$  (i.e.,  $h\psi$ , where  $h = 1$ ).

As stated in section (2), according to each data band range, we compute the area of the rectangle  $R_{x_{i-1}x_iC_iC_{i-1}}$ . For instance, in Figure 8, the segments belong to RDB, then the height  $\psi$  of the rectangle  $R_{x_{i-1}x_iC_iC_{i-1}}$  is equal to 34, and the area of the rectangle  $R_{x_{i-1}x_iC_iC_{i-1}}$  is equal to 34. The values of the area of the rectangle  $R_{x_{i-1}x_iC_iC_{i-1}}$  and the area  $D_i$  are presented in Table 1.

In our example, using Table 1, the area  $D_i$  belonging to Out-Of-Band, to RDB, or to DDB, is a decreasing function of the area  $A_i$  (cf. Figure 11).

By comparing the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_ix_ix_{i-1}}$  and the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_iC_iC_{i-1}}$ , the closeness of a segment of a subsequence to a data band range, namely RDB or to DDB, can be also speculated visually (Figures 7, 8, 9). The closer is the segment to DDB, the higher is its weight according to the whole sequence. Furthermore, in Table 1, by comparing the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_ix_ix_{i-1}}$  and the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_iC_iC_{i-1}}$ , the closeness of a segment of a subsequence to RDB or to DDB is also assessed. In Table 1, the lower is the area  $A_i$  of the trapezoid  $T_{f_{i-1}f_ix_ix_{i-1}}$  according to the whole sequence, and the higher is the area  $D_i$  of the trapezoid  $T_{f_{i-1}f_iC_iC_{i-1}}$  according to the whole subsequence, the lower is the weight of the corresponding segment according to the weight of the whole sequence. Then, the lower is the area  $A_i$ , the lower is the weight of the corresponding segment according to the weight of the whole sequence (cf. Figure 10).

**Table 1.** Area  $A_i$  of the trapezoid  $T_{f_{i-1}f_ix_ix_{i-1}}$  belonging to Out-Of-Band, RDB and DDB.

Area  $D_i$  of the trapezoid  $T_{f_{i-1}f_iC_iC_{i-1}}$  belonging to Out-Of-Band, RDB and DDB.

Area of the rectangle  $R_{x_{i-1}x_iC_iC_{i-1}}$  ( $h\psi$ ) belonging to Out-Of-Band, RDB and DDB. (cf. Figures 7, 8, 9).

	$A_1$	$A_2$	$A_3$	$A_4$	$A_5$	$A_6$	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	$D_6$	$h\psi$
OOB	16.5	18.5	19.5	21.5	23.75	24.75	8.5	6.5	5.5	3.5	1.25	0.25	25
RDB	27.5	31	32.75	33.75			6.5	3	1.25	0.25			34
DDB	37.6	42.8	42.35	38.45	36.2		9.4	4.2	4.65	8.55	10.8		47



For example [2], [4], in Table 1, the area  $A_1$  of the trapezoid  $T_{f_0f_1x_1x_0}$  belonging to *Out-Of-Band Data* has the lowest value (i.e. 16.5). Its corresponding area  $D_1$  of the trapezoid  $T_{f_0f_1C_1C_0}$  has the highest value (i.e. 8.5) according to  $D_2$ ,  $D_3$ ,  $D_4$ ,  $D_5$ , and  $D_6$  belonging to *Out-Of-Band*. We deduce that the segment  $f_0f_1$  belonging to *Out-Of-Band Data* has the lowest weight according to the weight of the whole sequence (cf. Figure 10). Then, the segment  $f_0f_1$  has the greatest distance to RDB (cf. Figure 7).

The area  $A_1$  of the trapezoid  $T_{f_0f_1x_1x_0}$  belonging to RDB has the lowest value (i.e. 27.5) of the RDB row in Table 1. Its corresponding area  $D_1$  has the highest value (i.e. 6.5) according to  $D_2$ ,  $D_3$ ,  $D_4$  of RDB (cf. Figure 8). Then, the segment  $f_0f_1$  belonging to RDB has the lowest weight according to the weight of the whole subsequence belonging to RDB. However, its weight is not the lowest according to the weight of the whole sequence.

The area  $A_2$  of the trapezoid  $T_{f_1f_2x_2x_1}$  belonging to DDB has the highest value (i.e. 42.8), and its corresponding area  $D_2$  has the lowest value (i.e. 4.2) according to  $D_1$ ,  $D_3$ ,  $D_4$ , and  $D_5$  belonging to DDB (cf. Table 1). Consequently, the segment  $f_1f_2$  belonging to DDB has the highest weight according to the whole subsequence. In other words, the segment  $f_1f_2$  has the nearest distance to DDB (Figure 9).

Using Table 1, the value of the area  $A_i$  helps to speculate the highest or lowest weight of a segment of a subsequence belonging to *Out-Of-Band*, to RDB and to DDB. The lower is the area  $A_i$ , the lower is the weight of the corresponding segment (cf. Figure 10). Furthermore, as previously described, using the area  $A_i$ , the weight of a segment can also be evaluated. Moreover, in our representation of time series, the importance of every segment of a subsequence can also be estimated using the corresponding graph of a subsequence. Our representation of time series makes possible to determine that if a segment belongs, for example, to the *Risky Data Band* (RDB), then its weight is lower than the weight of a segment belonging to the *Dangerous Data Band*.

#### 4. Conclusion and Outlook

Our representation of time series provides a new approach, which can be applied to different domains. One of its application is Temporal Web Mining (TWM). In the TWM process, the representation of time series data is taken into account. In our representation of time series, the idea of using data band ranges and assigning a type to each segment is new. This representation consists of two main types of data band ranges. The *Dangerous Data Band* (DDB) is the first data band range, and the second is the *Risky Data Band* (RDB). DDB includes the data during the occurrence of an important event, such as the values of a high wind speed. RDB consists of the data before a crucial occurrence (such as volcanic ash dispersal). To every segment, we assign a weight, which is evaluated based on a new approach that uses the data band ranges and areas. The closeness of a segment to a data band range can be a clue to find, to a certain extent, a crucial occurrence. Estimating the closeness of a segment to a data band range is considered by computing the area under a segment and comparing it to the area that separates it from DDB or RDB.

Future direction includes comparing the represented data in order to determine similar or frequent patterns. In our representation of time series, we used data band ranges and areas to determine the type and the weight of a segment. Adopting the idea of areas

is also important in discovering similar data features. By comparing the areas, different frequent patterns can be specified. The areas that occur frequently can be specified. The relationships between the areas can be studied in order to discover previously unknown data features.

## References

- [1] M. Samia, Temporal Web Mining, in *Proc. 15. GI-Workshop über Grundlagen von Datenbanken (Proc. 15th GI-Workshop on the Foundations of Databases)*, Tagermünde, Germany, June 2003, 27–31.
- [2] M. Samia, *Temporal Web Mining*, Shaker Verlag, August 2006.
- [3] M. Samia and S. Conrad, From Temporal Data Mining and Web Mining To Temporal Web Mining, *ser. Frontiers in Artificial Intelligence and Applications*, J. Barzdins and A. Caplinskas (Eds.), IOS Press, 2005.
- [4] M. Samia and S. Conrad, A Time Series Representation for Temporal Web Mining, in *Proc. of the 2006 Seventh International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2006)*, O. Vasilecas and J. Eder and A. Caplinskas (Eds.), Vilnius, Lithuania, July 2006, 132–140.
- [5] M. Samia and S. Conrad, A Representation of Time Series for Prediction, in *Proc. 22nd British National Conf. on Databases: BNCOD Data Mining and Knowledge Discovery Workshop*, D. Nelson, S. Stirr, H. Edwards, and K. McGarry (Eds.), Sunderland, UK: University of Sunderland Press, 2005, 109–116.
- [6] C. A. Kenedi, S. R. Brantley, J. W. H. II, and P. H. Stauffer, Volcanic Ash Fall: A "Hard Rain" of Abrasive Particles, U.S. Geological Survey Fact Sheet 027-00, [Online], Available: <http://geopubs.wr.usgs.gov/fact-sheet/fs027-00/>, 2000.
- [7] R. Kosala and H. Blockeel, Web Mining Research: A Survey, *ACM SIGKDD Explorations* **2**(2000), 1–15.
- [8] C. Antunes and A. Oliveira, Temporal Data Mining: An Overview, in *Proc. KDD Workshop on Temporal Data Mining*, San Francisco, 2001, 1–13.
- [9] W. Lin, M. A. Orgun, and G. J. Williams, An Overview of Temporal Data Mining, in *Proc. Australian Data Mining Workshop (ADM'02)*, Canberra, Australia, 2002.
- [10] T. Risch and L. Lin, Querying Continuous Time Sequences, in *Proc. VLDB'98*, Newport Beach, CA, USA, 1998, 170–181.
- [11] E. Keogh and P. Smyth, A Probabilistic Approach to Fast Pattern Matching in Time Series Databases, in *Proc. KDD'97*, Newport Beach, CA, USA, 1997, 24–30.
- [12] E. Keogh and M. Pazzani, An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback, in *Proc. KDD'98*, ACM Press, 1998, 239–241.
- [13] E. Keogh and M. Pazzani, Relevance Feedback Retrieval of Time Series Data, in *Proc. SIGIR'99*, Berkley, CA, USA, August 1999.
- [14] M. Samia, A Representation of Time Series for Temporal Web Mining, in *Proc. 16. GI-Workshop über Grundlagen von Datenbanken (Proc. 16th GI-Workshop on the Foundations of Databases)*, M. Samia and S. Conrad (Eds.), Monheim, Germany, June 2004, 103–107.
- [15] K. C. Heidorn, Beaufort Wind Speed Scale, [Online], Available: <http://www.islandnet.com/~see/weather/history/beauwsc1.htm>, 2004.
- [16] OFCM, Federal Meteorological Handbook No. 1: Surface Weather Observations and Reports, Office of the Federal Coordinator for Meteorological Services and Supporting Research, [Online], Available: <http://www.ofcm.gov/fmh-1/fmh1.htm>.

# The Framework for Business Rule Based Software Modeling: an Approach for Data Analysis Models Integration

Olegas VASILECAS and Aidan SMAIZYS  
*Vilnius Gediminas Technical University, Vilnius, Lithuania*

**Abstract.** This paper discusses a possibility to take business rules out of the software system code and proposes a method based on presented framework for use of such business rules for dynamical software system integrated model transformations into the final software system code or system requirements at the same time tracking relations between them through the entire business rule based information system engineering process.

**Keywords.** Business rules, model transformation, software engineering, data analysis, XML

## Introduction

Business system is a living organism and should support different living strategies, provide organizational structure with resources needed and produce synergetic benefit to get new resources used for expansion of the system and reduction of internal resource usage by implementation of data analysis based decisions, new technologies or optimization of organizational structure and business processes.

There are already large systems allowing automation of simple logic structured business processes but automation of less structured business processes with frequently changing logics without need of reengineering entire all the software system (SS) is still a problem. One of such processes is business system state based data analysis discussed in the presented paper.

Few reasons - difficulty of understanding of the scope of complex systems and catching all the logics and relations between rules describing business logics, limited time and no planned budget, usually force system designers and project managers to choose the easiest way and create new functionality instead of modifying or expanding the already existing functions. As the result the old functions are left without adaptation and modification. With the lapse of time it makes systems more and more complicated and finally available to be understood only by previous users. That is the main reason why new users are always shocked by the complexity of a new system.

In the paper we discuss a solution of such problems by extending engineering process and creation of a new framework, where all the models used in engineering process are divided into separate related models starting from the business system model and proceeding with information system model, software system model and finishing with generation and implementation of a new software system code or

implementation of such models directly into the final software system to allow dynamic executable code generation.

We are convinced that business rule driven dynamic data analysis and decision support according to the proposed framework can be used for creation of active software system automatically reacting to the changes of the business system state and modifying existing or creating new executable data analysis processes in the final software system code.

The paper is structured as follows: Section 1 discusses the related work; Section 2 introduces the framework for business rule driven software system design and the possibility of using the framework for data analysis software system design, using rule model driven transformations and dynamic business rule transformations. Section 3 introduces current state of our research in practical implementation of the proposed framework. Finally we present our conclusions and discuss future research in Section 4.

## 1. Related Work

The software system design according to the software engineering methods starts with writing requirements and specifications. Such specifications mostly are written in informal form to be understandable by business users. Following further implementation of such informal specifications it is very difficult to eliminate misunderstandings or misinterpretations and track all the changes needed in already implemented Software system when Business system changes.

Business systems are functioning according to business rules in specific business domain. Due to the dynamics of its nature, the business environment is changing frequently according to internal and external influences such as changes in law, new competition, etc. [1]. Business requires immediate and adequate reaction to the changes and immediate data analysis for decision support.

The changes of business environment rules are affecting business systems and the part used for data analysis and decision support first of all. Such systems through the time are integrating new features and functionality. This is determined by changes in business models, competition, legislation, business policy and rules.

Because of the need for immediate changes there is usually no time for adequate and detail analysis, system designers can not find where business rules have been implemented into the software system code, how they interfere with each other and which part of the code in the software system will be affected. Most of immediate changes are unexpected and have no assigned budget for them.

“Specifications understandable by users, developers and reusers must be simple and concise, must be abstract — no implementation, details must be complete — no gaps for developers to fill, must be precise — no guesses over ambiguities” [2]. Haim Kilov and other authors try to achieve it by embedding business rules into relations in graphical object oriented requirement specifications and models, creating reusable abstract constructs called business patterns.

Using this approach [2, 3] business rules are involved in capturing requirements and creating business or information system model and stored in graphical specifications. Although the main reason why business patterns or models are not reused without further manual interpretation is because the computation of a floating decimal number remained in gross-to-net pay obviously did not remain the same even in two installations in the same city. That means - design process does not eliminate or

control further manual transformation of the requirements and the final result usually depends on the developer. Thereby by implementing the requirement in the development process the relations to the business rules are lost. The other problem is that the meaning of the Business rule embedded in a graphical model can change already in design process by adding new business rules and storing them by adding new components and relations to the model. Such changes can result in contradictions and incompleteness of business rule set.

There are business activities where the business rules are being changed frequently and depend on some business parameters especially in data analysis and decision support. Most work in dynamic business service composition has focused on using work flows either as an engine for distributed activity coordination or as a tool to model and define service composition. Representative works are described in [4], where the authors discuss the development of a platform specifying and enacting composite services in the context of a workflow engine, using model driven service composition.

Analyzing the possibility of dynamic data analysis implementation authors usually mention two possible opportunities: active databases and warehousing with integrated On-line Analytical Processing (OLAP). In any case doing data analysis we do not modify the existing data model. We use different queries generating new views or generating data cubes in OLAP and slicing them from different perspectives as needed [5]. Data model in data analysis process is not changed. SS user can manipulate using data elements as well directly from DBMS, but business user's can not see how it corresponds to the business rules and business policy.

In [3] the authors state that by selecting and combining the most suitable and economical web services, business processes can be assigned dynamically by observing the changing business conditions. We think the same way can be used for dynamic data analysis process generation according to business situation. There are two different ways of data analysis process creation – design and storing of the different executable processes in software system, at the same time mapping different processes to different conditions stored in software system rules and the other one –storing business rules and transformations needed for data analysis process generation and compiling the process, using stored transformations, on the fly according to the current business rule set loaded in the Knowledge base of the system. This approach will be discussed in Section 4.

In [6] the authors analyze the basic elements in web service composition, present a rule driven mechanism to govern and guide the process of service composition in terms of five broad composition phases spanning abstract definition, scheduling, construction, execution, and evolution to support on demand business process selection for execution of the predefined business processes.

One of the assumptions all the standards for web service composition (e.g. BPEL - Business Process Execution Language) make is that the business process is pre-defined. Obviously this assumption does not hold if business needs to accommodate changes in applications, technology, and organizational policies. We agree with the authors that business processes can be dynamically built by composing web services if they are constructed based on and governed by business rules and we will expand this view creating framework for business rule driven software engineering in the next section.

## 2. The Framework

Traditional Information system engineering presented in Figure 1 is based on classical Waterfall Software system lifecycle and is dedicated for real world (business system) modeling using requirements, specifications and models usually embedding business rules and implementing them into the data, process or object models and the final software system (SS) code later on. Requirement engineering according to the traditional engineering from such perspective is technology independent and built on business rules and existing business infrastructure presented in figure as organizational structure. Later during the design period requirement model is transformed into technology dependent models and software code.

It is good for SS designed as tools allowing automation of business processes with simple predefined logics implemented into IF ... THEN, CASE logical sentences, cycles or procedure and function sequences. However it supports only slow changes in business systems without deep understanding how new SS will change the business system itself and without possibility of future modification without all the reengineering process.

In next section we introduce a framework approach to business rule driven data analysis software system design. We believe this approach will help to understand the issues, design activities and transformations needed to create infrastructure enabling to separate business rules and store them for further processing and transformation into different parts of final software system code and how business rules can be applied in each phase following the framework.

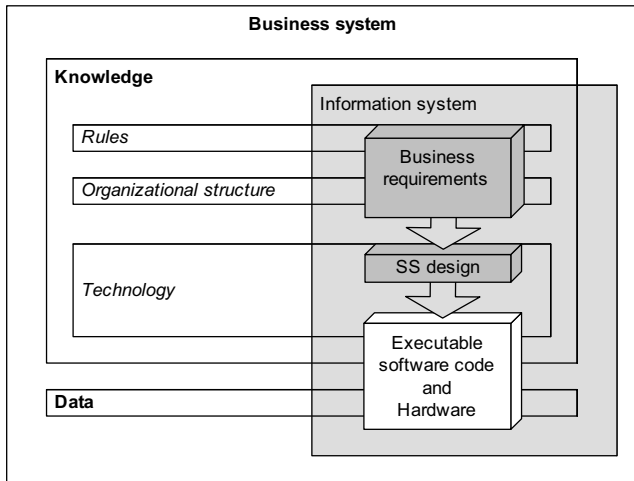
### *2.1. The Framework for Business Rule Driven Engineering*

If we look into business system differently as into a living organism it should support different living strategies, provide organizational structure with resources needed and produce synergetic benefit to get new resources used for expansion of the system and reduction of internal resource usage by implementation of data analysis based decisions, new technologies or optimization of organizational structure and business processes.

There are already large systems allowing automation of simple logic structured business processes and the question is how to automate less structured business processes with frequently changing logics without need of reengineering entire all the SS. One of such processes is business system state based data analysis.

To allow engineering of such dynamic data processes and explain the proposed approach we present a framework displayed in Figure 2. We look into system engineering as set of related models based on business requirements and model transformations, but the part of frequently changing logics is separated into new rule model and used entire engineering for transformation into other models or implemented directly into SS using information processing rules placed into the specially designed software component - business rule repository in the final Software system with a specific infrastructure described in [9] and discussed in Section 3.

Presenting the framework we suggest a new look into business rule based information system development process by introducing three system model architecture of the engineering process, where all the models created during development process are divided into separate system models starting from real business system, going further to the business system model, information system model, software system model.



**Figure 1.** Traditional software engineering

SS models are to be transformed into the final software system at the end of the engineering process, at the same time using model transformations in different abstraction levels e.g. process model (placed in the second row of the framework) from business process model to the information system process model, software system process model and final procedure source code in the software system.

Traditional engineering usually includes process, object and data models (displayed in the rows of the proposed framework). The entire traditional engineering process is done by using manual or semi automatic transformation of those models, but it excludes business rules. However every model in the business system is derived from business rules or affected by business logics embedded in such rules.

We suggest new Rule model column in the proposed framework to allow explanation of the separate processing of captured business rules. According to the proposed framework business rules are captured from real business system and stored in business rule model of the business system model. The business rules are acting in business system model and are used as a source for transformation into other models following framework going down into the related object, process and data business system models.

As mentioned before business rules can be transformed into information processing rules stored in the information system rule model, and data processing rules in the software system rule model and data processing software system code in the final software system.

All the parts of models in the framework are related and /or derived to/from each other, so the relations should be kept as much further as possible following transformations in the design process. The relations can be stored in specific design environment or repository and used for further modifications and reengineering doing what-if analysis to find affected parts of the participating models after the specific business rule is changed in the business system.

The framework proposes an ideal structure of the complete business rule driven software engineering process architecture.

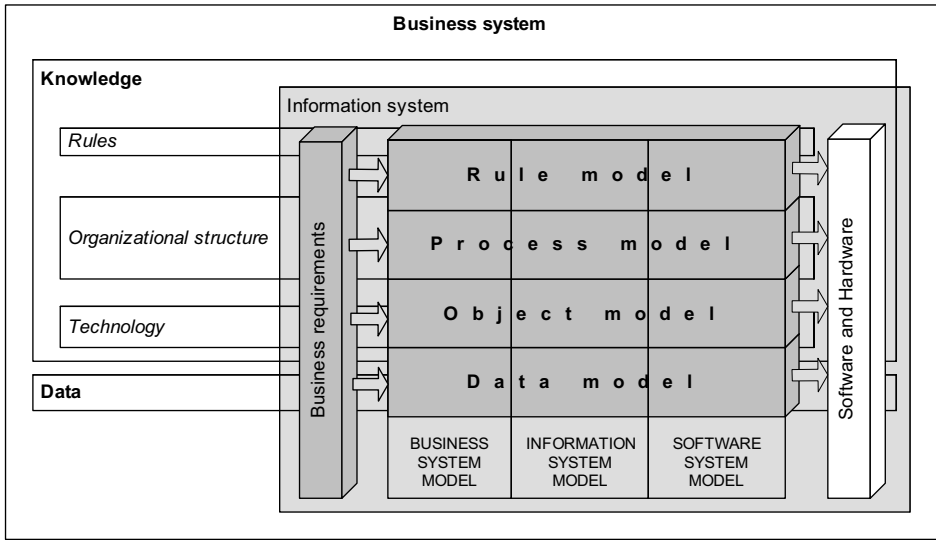


Figure 2. The framework for model transformation based software engineering

That is the main grid for better visualization of the engineering process using business rules. We do not mean that all the models should be preserved. We suppose this to be overloaded and sometimes impossible due to complexity, but we always need to choose the parts of the model which better suit a specific situation. The composition of such selected models and transformations is usually presented as a separate information system development method. If it could be ever a possible to realize fully automated model transformations and preserve all the relations we could get an ideal intelligent system creating and modifying itself by adding new business rules to the rule model and limited only by restrictions and availability of model structures used. Theoretically such system may generate several models according to the same rule set.

However absolute automation of information system development process is impossible due to the rules acting in real business system and not included into the models or even not suspected to exist at all.

The main difference of automated development processes and engineering processes done by humans is creativity. According to the currently existing facts (data) and knowledge (rule set) humans generate models and doing what-if analysis evaluate them by selecting the best using dynamically created testing criteria. Humans create models to simplify the real world and predict the result of real processes and relations between them included in the model.

Any automated process created to produce models should simulate such behavior. Unfortunately it is impossible for the current automated software design systems due to complexity and according to the traditional engineering all or the most part of the creativity is left for humans. In the next sections we present two different approaches of business rule based model transformation driven software system engineering according to the proposed framework: manual – where models are created and transformed by humans presented in Section 2.2 and dynamic moving part of the models directly into the final software allowing dynamic transformations and software



system operation based on automated logical derivations using business rules stored in the SS and facts representing current business system state described in Section 2.3.

## *2.2. Manual Business Rule Based Model Driven Information System Engineering*

The traditional engineering is covered by the framework. As an example it can be the engineering process starting from creation of Business system model, when we identify actors and business activities by creating business use case diagrams. By drawing data flow diagrams (DFD) we create information system process model in the information system model. We usually do business rule modeling too by creating business system and information system requirements in plain text (kind of informal business rule and information processing rule model) and embedding business rules in relations used in graphical diagrams [7]. Only part of the business process model is usually transformed into the information system process model and only part of the information system model is transformed to the software system model moving further according to the framework by creating UML activity and static structure diagrams, transforming part of them into the software model diagrams, data structures, interfaces and specifications later on. We always follow the framework: create rule models analyzing business logics and use business rule and system model transformations going through the system models. This does not depend on whether it is going in the mind of a designer, using graphical notation or on paper.

Business rules based model driven approach to the information system development described in [8] according to the framework can be explained as follows.

According to the proposed framework business rules should be defined precisely and declaratively and contain signature (operation parameters and their types), precondition (what should be true immediately before), post condition (what should be true immediately after), triggering condition (when it should be executed) and operations should preserve corresponding invariants (those properties of things and relationships that stay the same). The business rules are formalized in the information system model using XML and OCL languages and transformed to rule class diagrams and ECA rules. ECA rules are transformed into triggers in the active database management system (DBMS). Following rule transformations in parallel other models are created in the process, data and object model rows.

All the models included in the framework relate to each other and any modification in any of the model will fire modifications in the other models. Tracking of all the relations according to the proposed framework would empower simulation of such changes and enable identification of the affected parts of the models.

Data analysis software system design and a possibility of dynamic business rule transformations according to the proposed framework are discussed in the next sections.

## *2.3. Dynamic Business Rule Transformation Driven Software Design*

There are two main views in the dynamic business rule driven software system design. One of them is to design predefined executable processes and execute them by using rules in the software system, where processes and execution rules are derived from business rules using transformations [4]. The other one, discussed in our papers [9, 10], is where business rules and facts describing current business system state are loaded into inference engine of the software system and transformed into software system executable data analysis process according to the results of logical derivations.

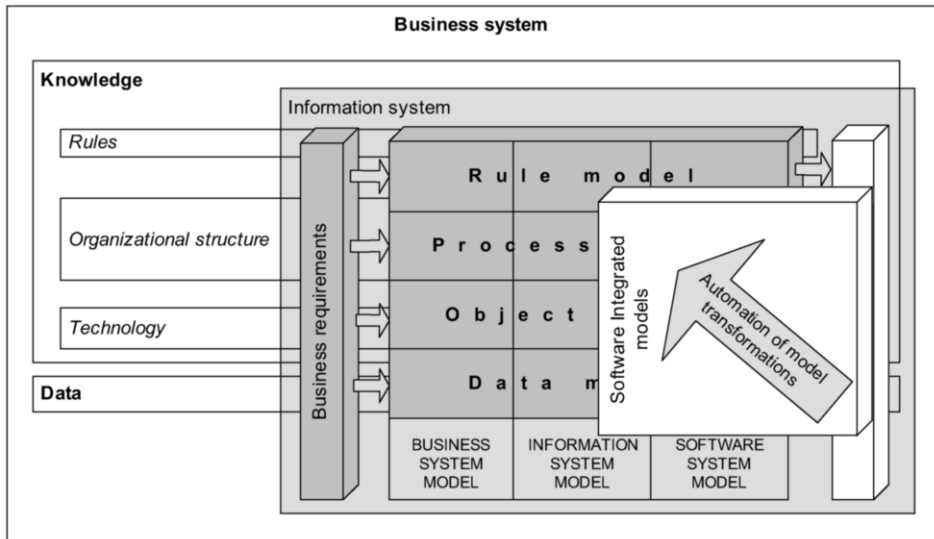
We have stated that the first view is not complete, because it is not enough to distinguish five broad phases defined in [3] (definition, scheduling, construction, execution and evolution) and create composition rules for data analysis system. A lot of system logics cannot be expressed by using separate rules. In most cases the result is more complicated and can be affected by combination of the whole rule set and interference of the rules. That is why we need to use more complicated logical derivation algorithms to evaluate the whole rule set before firing the first rule conformed to the predefined evaluation criteria at the predefined time according to the first view. On the other hand using the second view the result of the process depends on the inference machine used for logical derivations and in some cases it is impossible to predict a possible range of the results.

It is a problem because we need to have predefined transformations results or executable processes components for every derivation result. Anyway this problem can be solved building up new logical processing layer (e.g. decision processing subsystem) on the top of the software system to store part of the software integrated models and default transformations for processing of business rules stored directly in the SS and to generate SS code dynamically on demand according to the current business system state (Figure 3). We have proposed a method in [9, 10] for realization of such dynamic business rule transformations. It incorporates traditional design using modeling for analysts to model business rules using usual graphical representation in UML or represent the rules in formal XML form using predefined templates.

According to the proposed method programmers can create rule templates and transformations in XML language. The proposed method involves business users by adding special transformations created to translate formally specified business rules to human language allowing understanding of the business rules implemented in the software system processes. The BR representation templates and transformation schemas are the same for one class of the rules. Templates and transformation schemas can be reused many times and modified in entire all the systems at once.

The difficulty in creating classification schema is to allow management of such large set of business rules and possible transformations and combinations of rule sets needed. It is impossible to capture all the business rules and create models and transformations for them. It is needed to develop clear selection criteria during the engineering process and choose which part of the business rules will be transformed according to the requirements and implemented according to the traditional engineering process and which part will be stored in the software system for further derivations in the inference machine and dynamic transformation according to the rules in the information system model and transformed into triggers, stored procedures or software system code. Some of the business rules can be used as sources for paper documents describing business policy and business procedure logics and manuals for training employees.

According to the proposed method the rules are added to the rule model after logical processing in the inference machine to eliminate contradictions and prepare rule sets for further transformations. There is a theoretical possibility to dynamically create new information from data analysis results and translate them into new facts. These facts can be added to the knowledge base of the inference machine used in the rule model. It is possible to check if the loaded facts reflecting current business system state conform according to the business policy (business rules loaded in the knowledge base).



**Figure 3.** The framework used for software integrated model development

This would enable creation of intelligent integrated data analysis and business support system allowing dynamical business state evaluation and produce new knowledge or business rules. That would allow dynamical what-if analysis and decision modeling by eliminating contradictions in the knowledge base – changing business rules or adding new rules or facts for logical processing before implementing them into the business system. There is another challenge of automated discovery of new business rules from data analysis results using reasoning and data mining left behind the scope of this paper.

For implementation of the proposed method specially designed software system architecture is needed. It is possible to construct some of the models according to the framework in the mind of a designer or describe using informal language on paper, use graphical model visualizations or implement directly into the software. It is possible to do manual transformations like drawing UML diagrams according to the information system requirements or do semi or full automatic model transformations like creation of the database according to the database model diagram. Anyway it will fit into the framework. Comprehensive and consistent, models should be derived one from the other and related to each other. If we store and track the relations and model transformations we will be able to find which business rules and where are implemented in the final software system code and modify all the affected models and software system code by changing business rules and/or transformations in the Rule model stored in a specially designed repository.

### 3. Current State of the Framework Implementation Research

The proposed framework introduces an idea of using business rules for system evolution, managing business system changes by tracking business rules and changing all the dependent models and implementation into the software system.

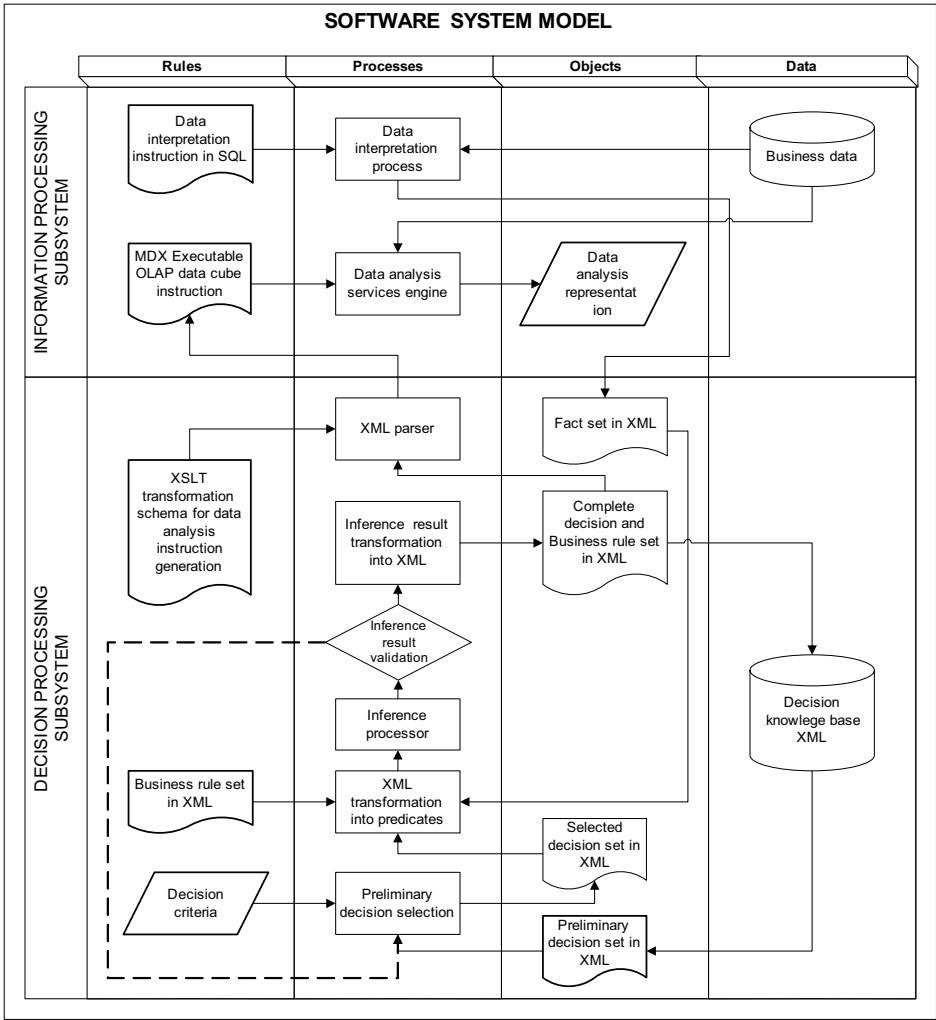


Figure 4. Example of integrated decision support and data analysis model software system architecture

The final ideal result of the framework could be a complete set of the models and their transformations, represented by different classes in the framework. Realization of fully automated model transformations preserving all the relations starting from the business rules could allow to get an ideal intelligent system creating and modifying itself by adding new business rules to the rule model and limited only by the restrictions and availability of model structures (classes) used.

Our investigations are based on XML transformations. We have used informal business rules captured in the business rule model of the business system model. The business rules are classified and formalized using DTD or XSD templates. Formal business rules are combined into different rule sets, depending on the expected transformation results and classes they belong to. In our experiments we have transformed rule sets into different UML models using XSLT transformation schemas

[10]. Such UML models were used for transformation into SQL triggers. Current automated computer-aided design (CAD) software systems allow the automated transformation of the software system UML models into software system components too. The latest research is oriented in examination of warehouse and OLAP technologies, possibilities of integrated model and business rule dynamic transformations into MDX instructions used to produce OLAP data cubes according to the model in Figure 4.

During our experiments we have discovered that going through the system models of the framework there is more and more automation possibilities of the transformations. It is because of more formalization and less uncertainty left due to the more detailed specification of the models.

Software system model transformations into the software system components are almost linear and can be produced using simple XSLT mapping between two different structures. However it is impossible to use such transformations in the initial system models. In the information system model and especially in the business system model there is a lot of rules acting at the same time and interfering with each other making contradictions and incomplete rule sets. Using traditional design methods contradictions are usually eliminated while using human creativity for development of new requirements or elimination of some of them out of the requirement set (rule set). Due to the large amount and complexity of the organizational structure of the requirements the contradictions are left and result in the inadequate implementations of the software system functionality. For solution of such problems and support for transformations we propose usage of dynamic business rule transformations using predicates and inference engine. This work is still in the research stage [10].

At this stage of the research we cannot fully replace traditional requirement engineering using proposed Rule model. However we have transformed business rules into the requirements using XSLT transformations. We will continue our research to extend and support the requirement engineering process by using business rule derivations using inference machine and allow creation of complete and non-contradiction requirements.

#### **4. Conclusions**

It is clear that the current system design methods are not capable of dealing with the complex and dynamic nature of business rules in a complicated and dynamic business environment. Several methods have been developed for the business rules driven engineering process, but there is no complete picture describing how and where the existing methods act in the whole engineering process.

The proposed framework introduces an idea of using Business Rules for system evolution, managing business system changes by tracking business rules and changing all the dependent models through the entire engineering process and implementation into software system. The framework introduces an ideal structure of integrated models for business rule driven software engineering process. It is too complicated to use all the included models. Therefore the framework models should be used according to the needs in the specific engineering process.

Tracking of the relations between different models and their sources in the framework allows following business rule transformations and implementations into the final software code. This allows easier implementation of the changed business

rules, redesign and modification of the models through the entire engineering process. The data collected during the design according to the proposed framework allow to create additional functionality easier by modifying the existing business processes affected by the changed business rules and replicate changes down to the software system changing the related existing software system processes instead of creating new and avoiding to get the system more and more complex.

We argue that it is very difficult to achieve fully automated business rule transformations to the software system components. However we have discovered that it is possible to achieve it in some parts of the software system by implementing specific business rule transformations into transitional models and final executable software system code used for intelligent adaptable data analysis.

Therefore the challenge is to provide a guide for further model transformation analysis and develop a method for business rule transformations into other models and final executable data analysis processes in an automated fashion. This can be allowed by creating a special software system architecture used to store all the transformations needed and execute them according to the current business system condition based on the dynamic data analysis results. Such software system could actively react and adapt to the changes in the business environment.

The work presented here summarizes the experience from business rule driven data analysis system design investigation. In the future we will further investigate issues such as automated business rule and rule model transformations using other artificial intelligence methods.

## References

- [1] B. von Hale. *Business Rules Applied: Building Better Systems Using the Business Rules Approach*. New York: John Wiley & Sons, Inc., 2001.
- [2] H. Kilov and I. Simmonds, Business patterns: reusable abstract constructs for business specification. In: *Implementing Systems for Supporting Management Decisions: Concepts, methods and experiences*, P. Humphreys et al., Eds. Chapman and Hall, 1996, pp. 225-248.
- [3] B. Orriens, J. Yang and M. P. Papazoglou, Model driven service composition. In: *Lecture Notes in Computer Science*, vol. 2910, Berlin: Springer-Verlag, 2003, pp. 75-90.
- [4] B. Orriens, J. Yang and M. P. Papazoglou, A Framework for Business Rule Driven Service Composition. In: *Lecture Notes in Computer Science*, vol. 2819, Berlin: Springer, 2003, pp. 14-27.
- [5] S. Mahajan, Building a Data Warehouse using Oracle OLAP Tools. Oracle Technical Report, *ACTA Journal*, September 1997.
- [6] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy and M. C. Shan, Adaptive and Dynamic Service Composition in eFlow. HP Lab. Techn. Report, HPL-2000-39. Available from: <http://www.hpl.hp.com/techreports/2000/HPL-2000-39.html>.
- [7] D. Rosca, S. Greenspan and C. Wild, Enterprise modelling and decision-support for automating the business rules lifecycle. *Automated Software Engineering*, vol. 9, no. 4, 2002, pp. 361-404.
- [8] S. Sosunovas and O. Vasilecas, Business rules based model driven approach in information systems development. In: Proceedings of the 18th International Conference on Systems for Automation of Engineering and Research (SAER-2004), R. Romansky, Ed. Sofia: CAD Research and Development Centre "Progress", Union of Scientists in Bulgaria, 2004, pp. 35-40.
- [9] O. Vasilecas and A. Smaizys, Business rule based knowledge base integrated intelligent system framework. *Information Sciences*, vol. 34, 2005, pp. 195-200.
- [10] O. Vasilecas and A. Smaizys, Business rule specification and transformation for rule based data analysis. In: Proceedings of 19th International Conference on Systems for Automation of Engineering and Research (SAER 2005), R. Romansky, Ed. Sofia: CAD Research and Development Centre "Progress", Union of Scientists in Bulgaria, 2005, pp. 35-40.

# IS Engineering

This page intentionally left blank



# Aspect-Oriented Use Cases and Crosscutting Interfaces for Reconfigurable Behavior Modeling

Lina NEMURAITĖ<sup>1</sup> and Milda BALANDYTE

*Department of Information Systems, Kaunas University of Technology, Lithuania*

**Abstract.** Aspect-oriented development has become one of the most intensively investigated themes in software development. In this paper, the method is proposed for reconfigurable modeling of aspect-oriented information system when <<Core>> and <<Aspect>> concerns may be represented separately and combined in different ways without changing their models or implementation. <<Aspect>> concerns are consistently represented during development process starting from <<Aspect>> use cases till crosscutting interfaces and templates for tailoring aspects for specific contexts. Examples from IT-Europe project are given where aspect-oriented concepts were used for modeling behavior of software agents performing self-management functionality of IT Knowledge Portal.<sup>2</sup>

**Keywords.** Aspect-oriented development, use case, crosscutting interface, subject-observer pattern, UML

## Introduction

The goal of this paper is a further elaboration and refinement of ideas presented in [1], where aspect-oriented use cases were proposed for extension of Design Independent Modeling (DIM) [2] of Information Systems having crosscutting concerns known as aspects [3, 4, 5]. In Design Independent Modeling, use cases are formalized using concept of object-oriented interfaces; aspect-oriented use cases are represented via crosscutting interfaces [6]. Crosscutting interfaces are different from usual (<<Core>>) interfaces as they affect associated interfaces via a dependency stereotyped as <<crosscut>> that changes (extends) the behavior of <<Core>> interfaces. Moreover, the <<crosscut>> dependency often is driven by aspect-oriented pattern, which slightly changes (adapts to the corresponding context) the behavior of <<Aspect>> interfaces.

It is desirable to keep core components and extensions (aspects) independent from each other making aspects reusable as much as possible. The effectiveness of aspect-oriented methods is conditioned by the possibility to introduce aspectual features (often they are additional behavioral features – so called “advices” in aspect-oriented terminology) in different points (“join points”) of a structure or the execution of a software system. In this paper, the method is proposed for reconfigurable modeling of

---

<sup>1</sup> Corresponding Author: Lina Nemuraite, Department of Information Systems, Kaunas University of Technology, Studentu 50-308, LT-51368, Lithuania; E-mail: lina.nemuraite@ktu.lt

<sup>2</sup> The work is supported by Lithuanian State Science and Studies Foundation according to Eureka programme project „IT-Europe” (Reg. No 3473).

aspect-oriented information system when <<Core>> and <<Aspect>> concerns may be represented separately and combined in different ways without changing their models or implementation (we do not devote here special section for definitions of aspect-oriented concepts as they were described in our previous work [1] and are thoroughly discussed in literature, e.g. [7, 8, 9]).

Aspect-oriented modeling is complicated, and despite the tremendous amounts of research by this theme there is no unified methodology how to deal with aspect-oriented development from initial requirements till implementation. The main contributions of this work are in:

- representation of aspect-oriented use cases;
- formalization of aspect-oriented use cases via crosscutting interfaces;
- using templates, configurations and defaults for composing aspectual interfaces with core ones.

Shortly, our objectives are in consecutive analysis process, as there is a lack of methodology for representing the initial aspect-oriented requirements in a consistent, formalized way for further transformation to design. The method is demonstrated with examples from Information Technology (IT) Knowledge Portal (investigated in IT-Europe project) – a Web information system having self-management aspects. Such systems react to external behavior not only by satisfying user requests but also by invoking additional services. In IT Knowledge Portal, the system agents calculate ratings, send reminders, assign reviewers, etc. Well-designed, such system may be fully capable automatically perform all desirable management and administrative activities, i.e., to automate work that is usually performed by system administrator or other responsible roles. The problem is in attaching/detaching management activities to various places of behavior of large variety of core components of the functioning system, as requirements for management may change. Event-based aspect-oriented approach seems to be the most appropriate solution for this situation, although it lacks smooth methodology going from requirements to code and often is hidden in technological level. In current paper, the attempts were made to represent aspect concerns in the early stage of development – from initial use cases till elaborated requirement level analysis model (DIM), ready for input to design phase. Even in this stage, aspect-orientation gives substantial benefits by crystallizing models and reducing huge amounts of specifications.

The rest of the paper is organized as follows. In Section 1, our approach to aspect-oriented use cases is explained. Section 2 presents configurable Subject-observer pattern for representation of interactions between core and aspectual interfaces obtained from use case specifications. Section 3 is devoted for separation of aspectual behavior, and Section 4 – for composition of core and aspectual behavior using templates, configuration of behavior and adaptation of aspects. In Section 5, related work is concerned. Finally, Section 6 draws some conclusions and highlights the future work.

## **1. Aspect-Oriented Use Cases**

Let's begin from example. In IT Knowledge Portal, users have different levels of ratings dependent from their activities and quality of these activities. The ratings are determined by users themselves through reviewing and evaluation. The user activities are: writing papers; announcing the hottest news; organizing conferences, presentations

and other events; making reviews, comments, evaluations, etc. Management of reviewing, notifications to users, paper acceptance and publishing, an evaluation of ratings is accomplished by system agents; there are a lot of similar though not fully identical activities. For example, reviewing is different for research papers and short papers – IT technology news, although they have many common features. It is desirable to design and implement such system in a way that similar parts may be reused and possibly modified in a future when the system evolves.

We do not consider here security, persistency, transactions and other mechanisms that usually are treated as aspects in aspect-oriented development (and in IT Knowledge portal as well). The attention is devoted to the management and time aspects that integrate with the rest behavior using events. A model and descriptions of use cases representing paper submission and reviewing are given in Figure 1 (the abstract pattern of such use cases was presented in [1]).

The idea to represent aspects as use cases in an early stage of development was borrowed from Jacobson [10], who has proposed to specify aspectual behavior via extension use cases. Such representation well conforms to extension semantics. However, Jacobson's <<Aspect>> use cases are dependent from <<Core>> use cases as they are added at concrete extension points. Moreover, Jacobson's use case specifications are directly used for implementation in AspectJ, omitting other modeling stages.

In our case, use case extensions are written independently from core use cases: extension points of core use cases are state-changing events serving as triggering events of <<Weaver>> use case, managing these extensions ("weaving" is aspect-oriented concept meaning integration of core and aspectual concerns). All extensions are written in specification of the <<Weaver>> use case. Invocation of additional behavior also is specified independently of <<Aspect>> use cases, using triggering events of <<Aspect>> use cases as extension points. In a general case, <<Weaver>> use case specification consists of pointcuts, expressed as constraints on possible join points, and additional behavior (advices) (here "pointcut" is an aspect-oriented term, expressing definition of a set of possible join points). In this paper, the changes of

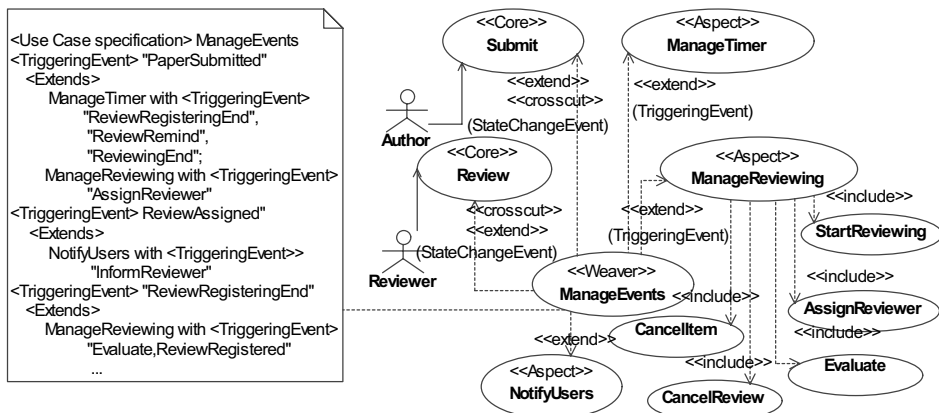


Figure 1. An example of aspect-oriented use cases

states of observable entities of problem domain are taken as <<pointcut>> constraints for discovering join points where additional behavior should be introduced. Pointcut constraints are represented using meta data structures that are considered in Section 4. Advices are represented as <<Aspect>> use cases at this stage.

## 2. Configurable Aspect-Oriented Subject-Observer Pattern

In Figure 2, the configurable Subject-observer pattern for interfaces representing aspect-oriented use cases is given.

This pattern is a concretized case of the pattern presented in [1], where choice of Subject-observer pattern and its employment for representation of interactions between <<Core>> and <<Aspect>> concerns was explored and compared with other approaches [11, 12, 13]. Event, SubjectRole, ObserverRole, AdviceOperation and Entity classes are introduced for enabling to configure management activities and other aspectual behavior invoked after observable events changing states of entities of problem domain (papers, reviews, users etc.). The “Default” class (the type of attribute AdviceOperation.default) is used for setting the parameters of aspects tailoring them for different cases (for example, a reviewing interval should be different for short news and full scientific papers). Having user interface for setting interfaces, events and entities that are playing roles of subjects, join points and observers in concrete cases of interactions between core and aspectual interfaces, the aspectual interfaces become configurable and adaptable for different core interfaces without changing their model or implementation.

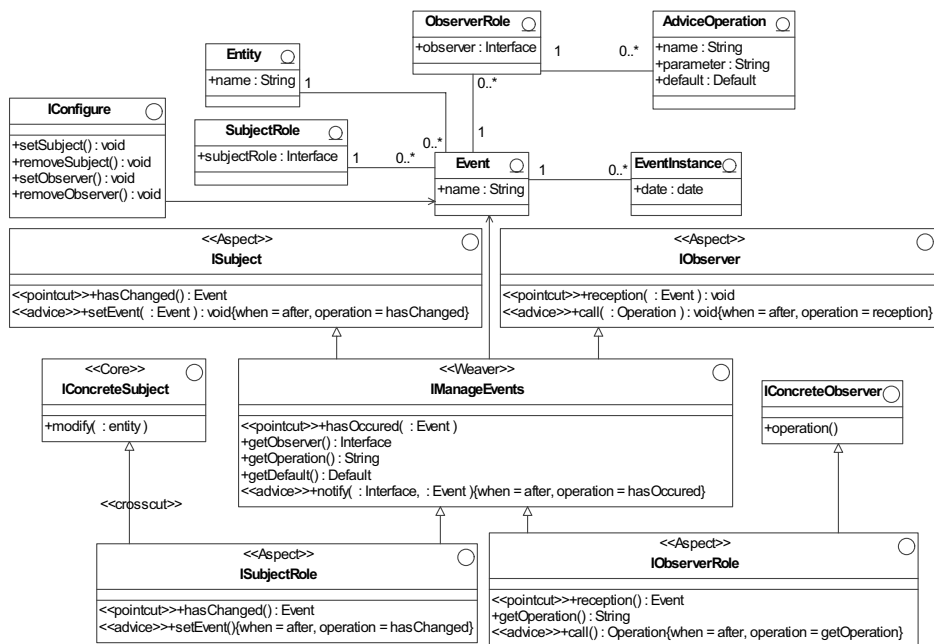
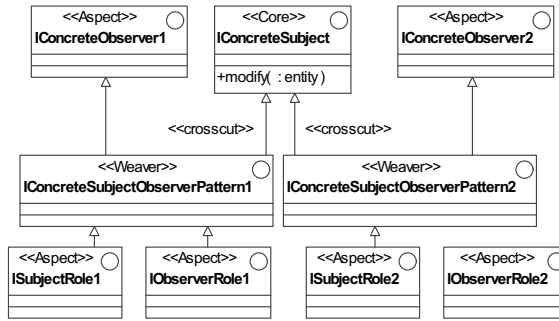


Figure 2. Configurable Subject-observer pattern



**Figure 3.** Concrete subject crosscut with several aspects

ISubjectRole and IObserverRole extend core or aspectual interfaces with crosscutting features, i.e., IConcreteSubject having several crosscutting interfaces will become extended with additional operations by a composition of these interfaces (Figure 3). Every concrete weaving of subject and observer may be represented as an individual use case and as a concrete Subject-observer pattern. We assume that aspects are “almost” domain-independent, therefore only specializations of core interfaces, playing roles of subjects, are marked by stereotype <<crosscut>>. The exceptions are specializations of aspectual interfaces playing roles of subjects, as the IManageReviewing interface in IT-Europe portal case (Figure 4) does.

Every interface, affected with crosscuts, must be extended with features, corresponding to crosscutting ones adapted to the context of subject-role interface, so specialization relationship stereotyped with <<crosscut>> is different from the convenient specialization. The tailoring of aspects for different domains is made by using Default class which attributes are parameters of the corresponding advice operation. Object diagrams representing event configuration and default parameters of advice operations for concrete domain are presented later in the paper (Figure 8).

For example, concrete interface ISubmitPaper will be extended with additional behavior originated from interfaces IManageEvents, IManageTimer, IManageReviewing and INotifyUsers (apart Security, Persistency and other concerns that usually are treated as aspects in aspect-oriented development). IManageEvents enables dynamic weaving of aspects by means of configuration: when particular event occurs (`hasOccured() = true`) IEventManager reads event recipients from ObserverRole objects (Figure 2) having links with corresponding event (an event is distinguishable from its instance that is called “eventOccurrence” in UML 2.0).

There are two fundamental ways to observing events: watching of state changes in database, or “reading” events together with their type information from runtime environment, using reflection [14, 15, 16]. The proposed pattern does not constrain the way for detection of events. The reflection usually raises the execution time. Currently, two prototypical implementations (in .Net 2) are made: the first one – using database events, and the second – creation of events explicitly.

Explicit creation of events narrows down the flexibility of the proposed approach. To make the system more adaptable, we must create events in advance, in all possible places where join points may be required. A large amount of created events will extra-load the weaver. The solution for this issue is to define operation for creation of events

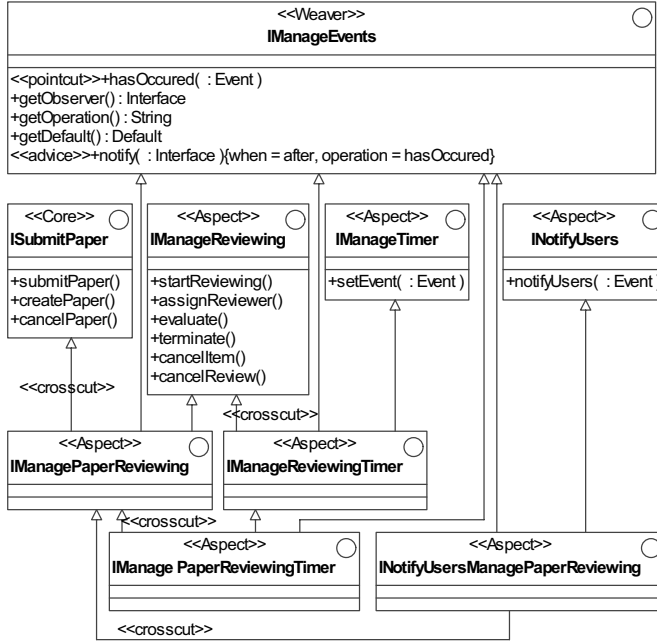


Figure 4. Paper submission interface extended with aspects

in such a way that event should be created only if it is indicated in Event configuration (Figure 2). The specification of operation for creation of events is given in OCL:

```

context ISubjectRole::setEvent(s:String) : EventInstance
pre: self.hasChanged(s:String) and Event → exist(e|e.name=s)
post: EventInstance → count(ei|ei.ocIsNew())=1
    and result.ocIsNew() and result.Event.name=s
    and result.date=now(),

```

where operation `hasChanged(s:String)` is defined as:

```

context ISubjectRole::hasChanged(s:String) : Boolean
result: Entity → exists(e|e.ocIsInState(s) and
    s.SubjectRole=self and
    not(e.state=e@pre.state))

```

Operations `hasChanged()` and `setEvent()` defined here are not complete self-sustaining operations as in general case [1]; they must be inserted in code after every operation that potentially may change states of entities. Additionally, if operation of `ISubject` may change states of several entities, operations `hasChanged()` and `setEvent()` should be inserted for state changing of every entity appearing as type of an argument of the corresponding operation of the interface.

Type of arguments of `hasChanged()` and `setEvent()` operations is a standard UML data type “String”. However, these arguments mean names of states of entities and names of events as names of states are used for naming of corresponding events. Hierarchy of names must be defined in system Namespace.

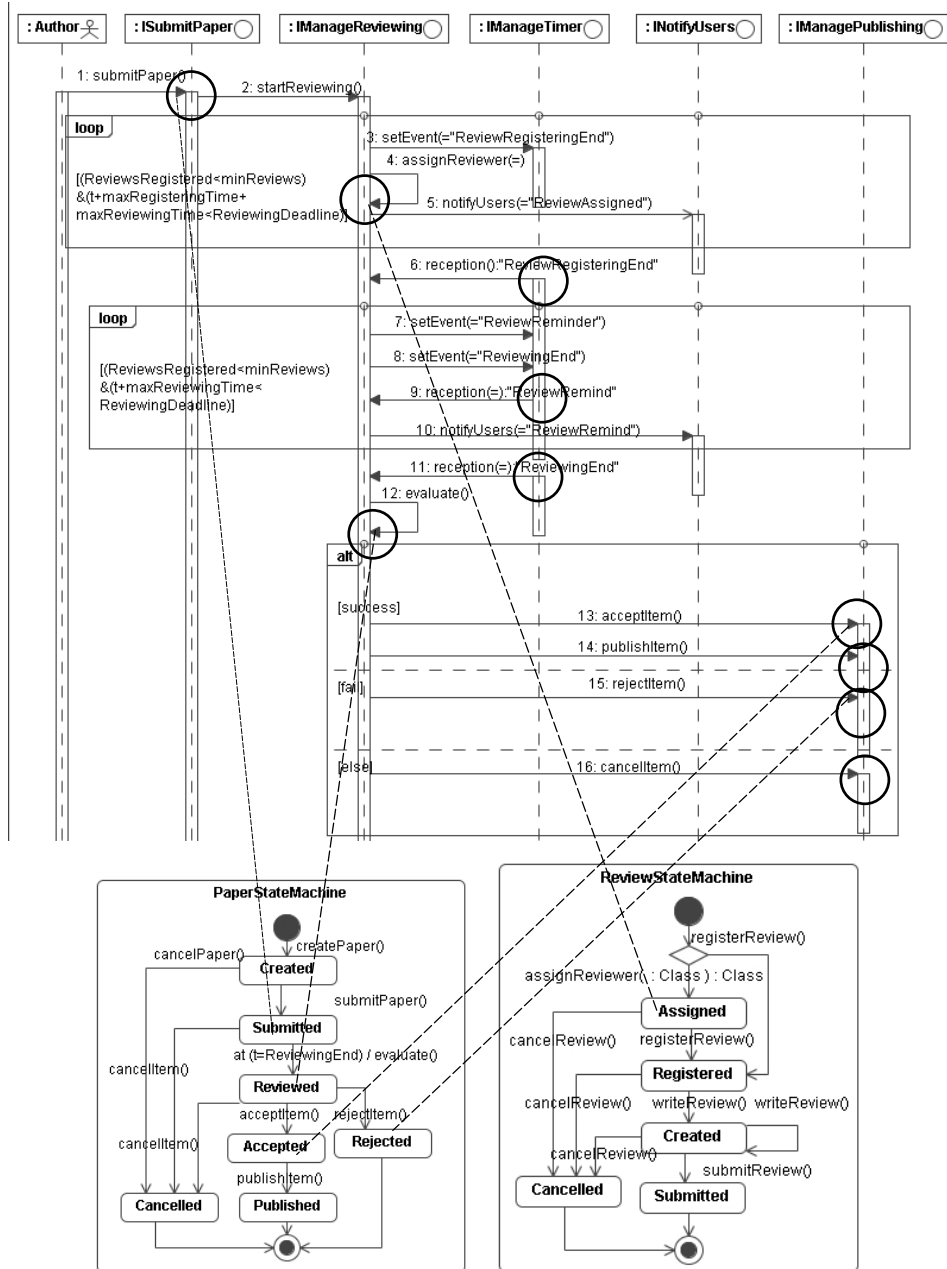
### 3. Interactions of Core Interfaces and Crosscutting Aspects

The behavior of aspect-oriented system must conform to business processes the system is intended to support. These processes often are described using activity or sequence diagrams consisting of activities and interactions of interfaces representing use cases. In Figure 5, a part of an initial sequence diagram describing submission and reviewing of a paper is presented. Suppose, we want to treat timer management, reviewing-management and notification concerns as aspects. The responsibility of *IManageTimer* would be a creation of all types of time-related events: instantaneous, time-interval events (delays and leads), and periodic ones. *IManageReviewing* crosscut with *IManageTimer* aspect must include formation of parameters relevant for creating of corresponding time-related events at desired join points. Recall that such join points are change events arising when entities are reaching some kinds of states that are desired to be managed. Such essential states are identified and specified during analysis of problem domain (state machines for *Paper* and *Review* entities are presented in Figure 5). Dependencies between state-changing events (circles) and state transitions are shown as dashed lines in Figure 5 (only few dependencies are shown).

Changing state of entity “*Paper*” to “*PaperSubmitted*” invokes a set of activities performed by system agents. Time events are set by the *TimerManager* agent, implementing the *IManageTimer* interface. These time events are: the end of registering reviews, reminding about the end of reviewing, and the reviewing itself. Also, reviewers are assigned by the *ReviewingManager* agent, implementing *IManageReviewing* interface; and, finally, assigned reviewers are notified by the *Notification* agent about the proposals to revise assigned papers.

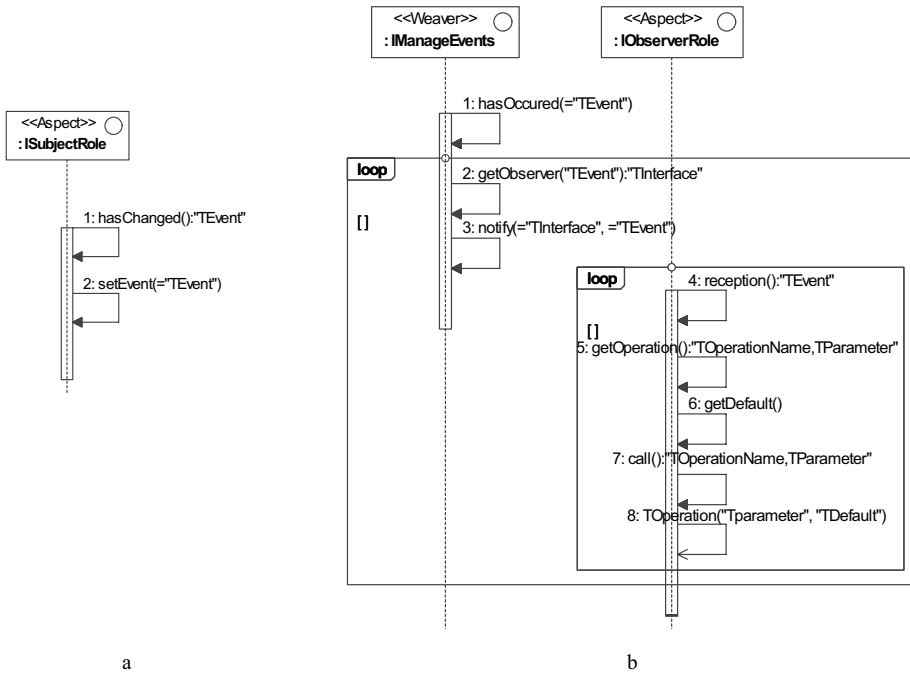
In ordinary Object-oriented design, such invocations will be performed by defined sequence of message-sending actions. For changing this sequence (adding or deleting actions, changing the order or implementations of actions), the corresponding design or implementation should be changed. In proposed configurable Aspect-oriented development, where core and aspectual interfaces are separated, it is possible to change desired actions by setting them by *IConfigure* interface (Figure 2). The generic sequence diagram now looks like one in Figure 6.

The interactions between interfaces, playing roles of subjects, and interfaces, playing roles of observers, are implemented by setting events on an event queue and “reading” them from the queue (operation “*hasOccured()*”). For assuring these interactions, all observable state changes must be complemented with creation of events, and all desired additional behaviour (“advices”) should be declaratively specified as operations of crosscutting interfaces of observers. That is, desirable scenarios of Information System must be specified using *Event*, *SubjectRole*, *ObserverRole*, *AdviceOperation* objects (Figure 2). Furthermore, Default objects must be created for those Subject-observer patterns that require default values for performing crosscutting operations. For example, reviewing management requires to know what time interval is committed for registering for reviewing and reviewing itself; what maximum time is devoted for prolongation of reviewing, etc. These defaults may be different for different reviewing subjects: for example, full papers require minimum three reviews and reviewing time is two months; and news require smart revision (1 hour) and only one review with conclusion “Yes” or “No”. Default and Parameter are different in that Default values are the same for all instances of particular event, when Parameter values are different.



**Figure 5.** Sequence diagram for Use Case “Submit Paper” and state machines of entities “Paper” and “Review”





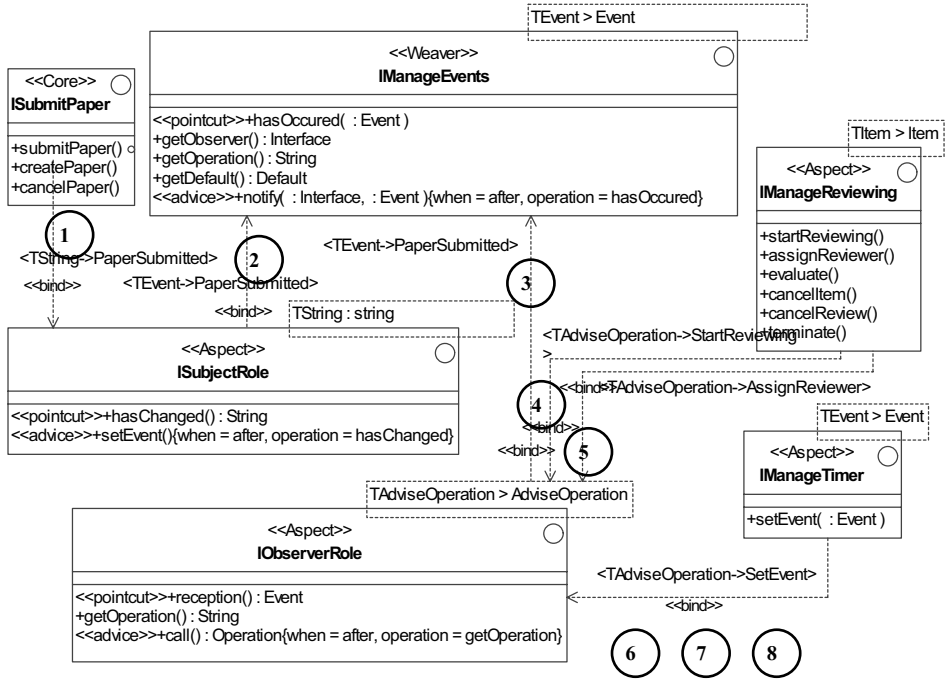
**Figure 6.** Sequence diagrams of aspectual interfaces ISubjectRole (a), IManageEvents and IObservableRole (b)

#### 4. Using Crosscutting Interface Templates for Aspect Weaving

In proposed pattern in Figure 2, multiple inheritances between interfaces were abundantly used. Authors of [17] argue, that multiple inheritances or mixins [18] are replaceable by templateable elements (at our abstraction level, such elements are interfaces). Instead of creation of multiple extensions, it is desirable to use bindings of templates representing aspect patterns (Figure 7). Nevertheless, creation of these bindings requires a lot of thorough work – it is desirable to automate the bindings using domain analysis artefacts, or at least to enforce a strict design discipline.

The order of bindings of templates is marked with numbered circles. ITimer operation “setEvent()” is bound 3 times; all bindings are specified as metadata using classes Event, ISubjectRole, IObservableRole, Entity, AdviceOperation, and their parameters. The example of such specifications is given in Figure 8, where the same scenario as in Figure 5 is encoded in objects.

Part of specifications in Figure 8 may be derived automatically from sequence or activity diagrams, representing business processes, or, conversely, sequence diagrams may be constructed from objects using transformations between object and interaction metamodels. The main constraints added to these metamodels are strict usages of names, and the same names are used for states of entities and events, playing roles of join points where additional operations are introduced.



**Figure 7.** Binding aspect templates for event “PaperSubmitted”: core interface ISubmitPaper (playing role of subject) is bound to template IManageEvents that in turn binds to templates IManageReviewing and IManageTimer; the bindings are managed by objects specified using IConfigureInterface (Figure 2). It is possible to change review management and other aspects by changing the configurable objects

Another strict requirement is raised for specification of default values for parameters of advice operations. Default parameters are configurable, but every adoption of aspect template requires to setting of defaults. If some default is not specified for an individual case, the nearest more general default is used. Default classes are different for every aspect or operation, and default objects are different for every usage of them. The example of TimeEventDefault class and objects is given in Figure 8.

## 5. Related Work and Discussion

This method is build on the basis of several proposals. Besides DIM [2] modeling and verification, the ideas of Jacobson method [10] are used, but differently from Jacobson, aspectual use cases are independent of base behavior, described in core use cases. We extend the use of UML stereotypes proposed in [19] to specification of use cases, so aspectual concepts are applied from the very initial phase of development.

For verification of aspect-oriented models, DIM verification [2] is used extended with composition of sequence diagrams representing interactions of core and aspectual use cases (the similar method is proposed in [20], although our DIM transformations

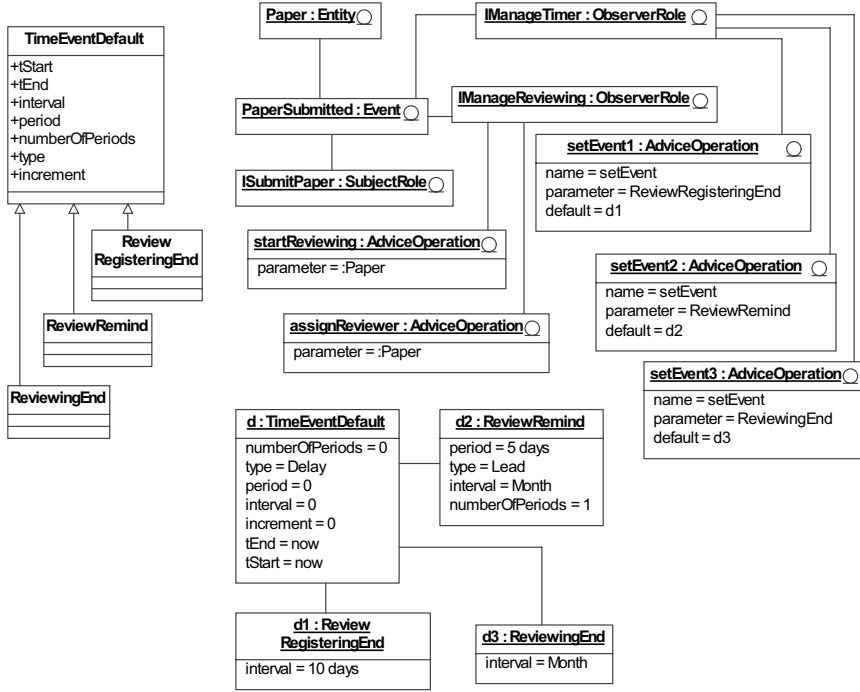


Figure 8. Objects, representing configuration for observed event “PaperSubmitted”

between sequence diagrams and state machines are bi-directional and have different semantics).

The Observer pattern is a fundamental pattern in aspect-orientated development, integration environments and event-based architectures. In aspect-oriented design, Subject and Observer are replaced with Subject-observer pattern [11, 21, 22, 23], where concrete subject and concrete observer are only logically related; inheritance is replaced with role modeling.

We have taken the [18] principles as theoretical basis to represent aspects as mixins; although for practical representation the configurable binding [17] adopts features of design patterns and mixins combining the advantages of both strategies, enabling reusability of implementation as well as reduces performance and resource consumption of the system. In our approach, metadata are proposed for configuration; aspects are tailored to core elements using default objects. Our proposal uses meta-level, model-level and instance-level concepts to represent design-independent, aspect-oriented model in terms of abstract interfaces and entities of problem domain.

In Model-driven development, specifying transformations between aspect-oriented models at various levels of abstraction can be a complex task as system features are tangled with other system features and elements to be transformed are distributed across a model. Although, aspect-oriented concepts are incident to Model-driven principles as transformation, composition and weaving are very similar in their nature [24, 25, 26, 27, 28, 29]. We use [2] transformations and some transformations, similar to [26, 27]; although for our overall purpose the additional transformations are needed.

## 6. Conclusions and Future Work

Aspect-orientation is a perspective but not an easy way and requires more efforts for achieving the higher quality of models. The proposed method covers relatively a narrow scope of the whole problem of aspect-oriented development, though it fits to a large variety of information systems concerned with management of business processes, self-managing software agents and time events. It is directed mainly to dynamic weaving of core functionality and aspects and is based on event modeling and aspect-oriented Subject-observer pattern, but static weaving also is not avoided.

The method is presented using examples, although it is based on ontologically founded principles of conceptual models, addressing the multiple inheritances and role modeling. The use of aspect-oriented separation of concerns leads to better design ensuring extensibility, adaptability and reuse. These shortcomings of object-oriented methodologies are intuitively understood by programmers, developers and even business analysts dealing with early stages of development.

The main contribution of the work is representation of crosscutting concerns by aspect-oriented use cases and mapping them to crosscutting interface templates. Adapting aspectual interfaces to different contexts is performed by binding core interfaces to aspect templates by setting template parameters and advice operations defaults. Required configurations of core and aspectual interfaces are managed by configuring metadata of weaving scenarios. We could not completely avoid “invasive” impact on core interfaces when applying aspects – in current implementations, events should be explicitly created in places of potential join points. Although, in configured application, only observable events will be created.

The proposed method requires of the strict development discipline, strong adhering to naming scheme etc. The real effect of this methodology may be achieved only applying Model-driven development principles. According to our previous proposal, scenarios of weaving core and aspects should be checked using automatic or semi-automatic procedures of generation of sequence diagrams and bi-directional transformations between sequence diagrams and state machines for verification. Besides ensuring correctness of aspect-oriented models, currently we are working on automation of static weaving for introduction of advices caused by ISubjectRole crosscuts, and capabilities to automate setting of configurations from initial business process models.

## References

- [1] M. Balandyte and L. Nemuraite. Time, event and self-management aspects in model-driven development of information systems. In: *Proceedings of the 7th International Baltic Conference on Databases and Information Systems*, Vilnius, Lithuania, 2006, 151 – 158.
- [2] L. Ceponiene and L. Nemuraite. Design independent modeling of information systems using UML and OCL. In: J. Barzdins, A. Caplinskas (eds.), *Databases and Information Systems*, Selected Papers from the 6th International Baltic Conference DB&IS'2004, IOS Press, 2005, 224-237.
- [3] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, Ch. Lopes, J. Loingtier, and J. Irwin. Aspect-oriented programming. In: *Proceedings of 11th European Conference on Object-Oriented Programming (ECOOP '97)*, LNCS **1241**, Springer-Verlag, 1997, 220–242.
- [4] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W. G. Griswold. An overview of AspectJ. In: J. Lindskov Knudsen (ed.), *ECOOP 2001 - Object-Oriented Programming*, LNCS **2072**, Springer, 2001, 327-355.

- [5] C. von Flach, G. Chavez, and C.J.P. Lucena. A theory of aspects for aspect-oriented software development. In: *Simpósio Brasileiro De Engenharia De Software*, Manaus: Editora da Universidade Federal do Amazonas, 2003, 130-145.
- [6] W.G. Griswold, M. Shonle, K. Sullivan, Y. Song, N. Tewari, Y. Cai, and H. Rajan. Modular software design with crosscutting interfaces. *IEEE Software* **23**(1) (2006), 51 – 60.
- [7] A. M. Reina, J. Torres, and M. Toro. Towards developing generic solutions with aspects. In: *Proceedings of 5th Aspect-Oriented Modeling Workshop (AOM) in Conjunction with the UML 2004 Conference*, Lisbon, Portugal, November 2004.
- [8] S. J. Mellor. A framework for aspect-oriented modelling. In: *Proceedings of 4th International Workshop on AOSD Modelling with UML*, San Francisco, CA, USA, October 2003.
- [9] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, and M. Wimmer. Towards a common reference architecture for aspect-oriented modeling. In: *Proceedings of 8th International Workshop on Aspect-Oriented Modeling, in conjunction with AOSD'06*, Bonn, Germany, March 2006.
- [10] I. Jacobson and Ng. Pan-Wei. *Aspect-Oriented Software Development with Use Cases*. Addison Wesley Professional, 2004.
- [11] E. K. Piveta and L. C. Zancanella. Observer pattern using aspect-oriented programming. In: *Scientific Literature Digital Library: CiteSeer.Ist*, 2003.
- [12] P. Th. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys* **35**(2) (2003), 114-131.
- [13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Pearson Education, 1994.
- [14] S. Chakravarthy and S. Varkala, S. Dynamic programming environment for active rules. In: *Proceedings of the 7th International Baltic Conference on Databases and Information Systems*, Vilnius, Lithuania, 2006, 3-16.
- [15] S. Herrmann, C. Hundt, and K. Mehner. Mapping use case level aspects to ObjectTeams/Java. In: *Workshop on Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design, in Conjunction with the OOPSLA Conference*, Vancouver, Canada, October 2004.
- [16] W. Schult and A. Polze. Aspect oriented programming with C# and .NET. In: *Proceedings of International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, Crystal City, VA, USA, 2002, 241-248.
- [17] S. Apel, H. Sichtung, and K. Böhm. *Configurable Binding: How to Exploit Mixins and Design Patterns for Resource-Constrained Environments*. Technical Report No. 14/2004, Department of Computer Science, Otto-von-Guericke University, Magdeburg, Germany, December 2004.
- [18] G. Guizzardi, G. Wagner, N. Guarino, and M. van Sinderen. An ontologically well-founded profile for UML conceptual models. In: A. Persson, J. Stirna (eds.), *Advanced Information Systems Engineering*, Proceedings of the 16th International Conference CaiSE'2004, LNCS **3084**, Springer, 2004, 112-126.
- [19] O. Aldawud, T. Elrad, and A. Bader. UML profile for aspect-oriented software development. In: *Proceedings of 3rd International Workshop on Aspect-Oriented Modelling* held with the AOSD 2003, Boston, USA, March 2003.
- [20] J. Whittle and J. Araujo. Scenario modelling with aspects. *IEE Proceedings - Software* **151**(4) (2004), 157-172.
- [21] J. Hannemann and G. Kiczales. Design pattern implementation in Java and AspectJ. In: *Proceedings of 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications: OOPSLA'02*, Seattle, Washington, USA, 2002, 161-173.
- [22] C. von Flach, G. Chavez, A.Garcia, U. Kulesza, C.S. Anna, and C. Lucena. Taming heterogeneous aspects with crosscutting interfaces. In: *Proceedings of the ACM Sigsoft XIX Brazilian Symposium on Software Engineering (SBES'05)*, Uberlandia, Brazil, October 2005.
- [23] A. Garcia, U. Kulesza, C. Chavez, and C. Lucena. The interaction aspect pattern. In: *Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP2005)*, Isree, Germany, July 2005.
- [24] J. Bézivin. From object composition to model transformation with the MDA. In: *Proceedings of TOOLS USA*, Santa Barbara, USA, August 2001.
- [25] M. Didonet del Fabro, J. Bezivin, F. Jouault, E. Breton, and G. Gueltas. AMW: a generic model weaver. In: *Proceedings of 1ère Journée sur l'Ingénierie Dirigée par les Modèles (IDM05)*, Paris, 2005.
- [26] R. France, I. Ray, G. Georg, and S. Ghosh. An aspect-oriented approach to early design modelling. *IEE Proceedings - Software* **151**(4) (2004), 173-185.
- [27] D. Simmonds, A. Solberg, R. Reddy, R. France, and S. Ghosh. An aspect oriented model driven framework. In: *Proceedings of 9th International IEEE EDOC Conference EDOC'2005*, Enschede, The Netherlands, Sept. 2005, 119 – 130.

- [28] I. Krechetov, B. Tekinerdogan, A. Garcia, C. Chavez, and U. Kulesza. Towards an integrated aspect-oriented modeling approach for software architecture design. In: *Workshop on Aspect-Oriented Modeling* held with the AOSD'06, Bonn, Germany, March 2006.
- [29] B. Baudry, F. Fleurey, R. France, and R. Reddy. Exploring the relationship between model composition and model transformation. In: *Proceedings of Aspect Oriented Modeling (AOM) Workshop Associated to MoDELS'05*, Montego Bay, Jamaica, October 2005.

# Integrated Enterprise Information Systems: Thinking in Component Concepts

Audrone LUPEIKIENE  
*Institute of Mathematics and Informatics  
Akademijos 4, Vilnius*

**Abstract.** Current component-oriented theory does not provide adequate support for information systems development. To say more precisely, it is forming. The main purpose of this paper is to discuss the attempts to combine classical enterprise information systems development methodologies with component-based approach. The paper advocates and argues for the use of component-based paradigm for the development of systems at all three layers of enterprise. It presents concepts and principles of this paradigm in enterprise engineering context. The integrated enterprise information system components, their structure and types are discussed and defined.

**Keywords.** Information system component, component-based paradigm, integrated enterprise information system

## Introduction

There is a strong trend towards higher reuse in software engineering. Component-based approach [1, 2, 3, 4] significantly lowers the barriers for reuse of prebuilt self-contained blocks in developing of large-scale software systems. Assembling systems from existing components has already proved its importance and value in many kinds of older engineering subdisciplines, including civil, naval, mechanical, aerospace engineering. It should be noted that in hardware engineering component-based approach is widely used, too. For the years, producing of computer hardware has radically changed. However, the state of affairs in software development cannot be characterized by such fundamental changes. Although research was very fruitful in component-based software engineering during recent years [1, 2, 3, 4, 5, 6, 7], there still are open questions and problems to solve: system modeling, analysis and design methods in the component-oriented manner, proper component modeling notations, architectural theory. Much effort and attention was primarily focused on software component technologies and platforms, and the industrialized software production is nearly achieved.

Nevertheless, current component-oriented theory does not provide adequate support for information systems development. Component-based paradigm, which supports interface-oriented development and focuses on the interactions of autonomous components, is a quite new paradigm in this context. To say more precisely, the paradigm is forming. Thus, the ideas presented should be developed or denied in near future.

This paper discusses the attempts to combine classical enterprise information systems development methodologies with component-based approach, i.e. extending component thinking to information systems development. The paper presents the hypothesis that integrated enterprise information systems (IEIS) can be conceptualized and developed as component-based systems. This implies that concept of component is significant and plays an essential role throughout the information system development lifecycle, not only in the software development process, resulting in binary or source code packages. The hypothesis is based on research done in the field of unification of approaches used to model business processes, information systems (IS) and software systems and to create an environment that facilitates the development of the whole enterprise system [8, 9, 10]. The objective of this paper is to advocate and argue for the component-based IEIS development paradigm and to offer the answer to the question “what is IEIS component”.

The rest of the paper is organized as follows. Section 1 explains our start position – the conceptual framework that conceptualizes business, information and software systems in uniform manner. Section 2 discusses and provides arguments for the component-based approach in the context of enterprise systems engineering. Section 3 defines the integrated enterprise information systems components. Related works are discussed in Section 4. Finally, in Section 5 we present concluding remarks.

## 1. Enterprise System and its Integrated Information System

A variety of approaches and multiplicity of different terminology sets are in use by different developers of business, information and software engineering systems. We will present a short explanation of our conceptual framework to clarify our start position.

An enterprise is a three-layered system consisting of business systems, information systems, and supporting software systems [8]. Any information system is a constituent part of business system and any software system is an essential part of a particular IS. Enterprise information system *is integrated* if and only if the lower-level systems are aware of higher-level systems and the lower-level systems are constrained by rules governing processes in higher-level systems (Figure 1); in other words, all three layers are properly integrated. So, integrated enterprise information system is developed on the base of business system model. The entities (objects, processes, events, etc.) presented in the business model map to entities in information system. However, this is not a one-to-one mapping. An entity in a business model cannot always be translated into one entity in information system, there are some entities in a business model that are not present in information system, and vice versa.

We thought of software system as consisting of software programs (the executable parts of a software system), files, databases, etc., designed to perform a well-defined set of tasks or to control processes and equipments. A set of hardware is necessary to execute a software system. So, proper sets of functional entities capable to realize functional operations (manageable units of work) must be defined at the information technology, as well as at the information processing level. We strongly separate organizational structures and business systems [8]. It should be noticed, that we do not deal with development of organizational structures at the business level. Functional entities at the information processing level are defined by the services they deliver, i. e.,



we deal only with the roles – a set of certain behavioral rules associated with a certain organizations or posts.

It seems that the scientific community agree on the definition of enterprise system as consisting of three-layers, see, for example [11, 12, 13]. Developers of IEIS start with understanding of the purposes to be served by those systems to enable business to achieve its goals. They are concerned with the content and usage of IEIS. Of course, developers create the technology-based information systems. It is the second step, in which they consider technology; thus this step concerns more technical issues. However, during the development of computerized systems to support business systems, the middle information processing level (information systems in a broad sense [8]) as a rule “disappears”. All the attention is given to the support of business processes by applications and to the support of applications by software platform and networking hardware. In other words, after the analysis of business system and its needs the attention is directed to the development of *integrated software system*. To illustrate such a situation, let us consider the Enterprise Resource Planning (ERP) systems that address the needs of all organizational units of an enterprise. The arguments are these:

- It is declared that ERP systems take a holistic view of information across the organization (see, for example [14, 15]). However, ERP definitions present these systems as package software solutions [15], i. e. ERP are thought of as generic software systems tying together and automating traditional back office functions [16].
- ERP implementation is treated as software project, which brings new patterns of business and new patterns of technology to the organization. Typically, ERP implementation involves business process reengineering (as well as information processing reengineering, which by default is included in business process reengineering), ERP software customization, configuration, and installation.

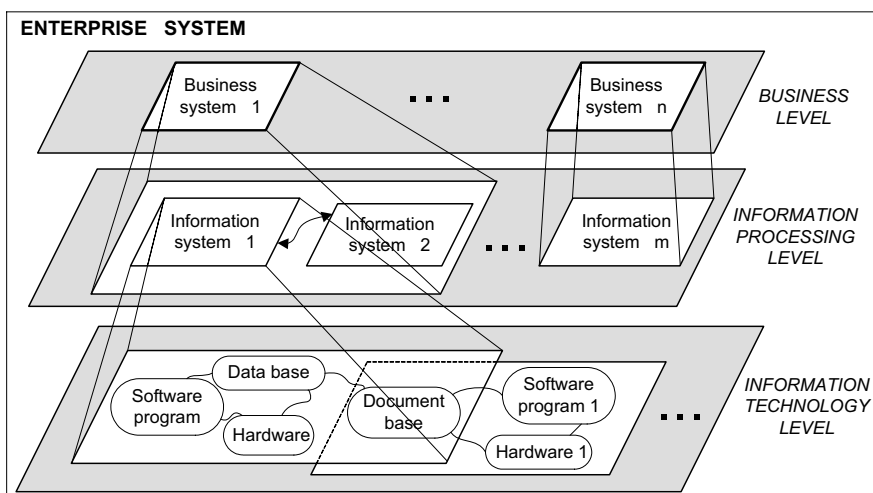


Figure 1. The three-layered enterprise system

Integrated enterprise information systems (IEIS) are very sophisticated socio-technical systems. This fact implies the following:

- IEIS (i. e., information processing system – computerized or not) as the whole should be developed, and its development should be treated as engineering discipline. In other words, analysis, design, construction of IEIS is realized using scientific principles, paradigms, methodologies, and methods.
- IEIS should be able to respond quickly to changes in business requirements as well as to technology changes. As a result of this, the development methodology should enable reuse at the information processing level, flexible IEIS modification, and the complexity management.

## 2. Component-Based IEIS Development Paradigm

### 2.1. Rethinking the Concept of Component

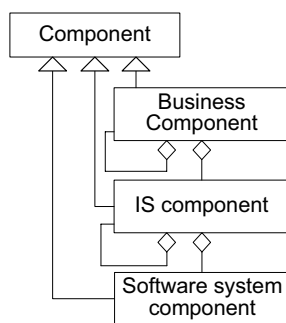
There exist various definitions of a component [1, 2, 3, 5, 6, 12, 13, 17, 18] showing that authors use components differently. The various definitions are proposed to focus on the different aspects of a component, depending on what one considers the most important feature. On the other hand, different component definitions (e.g., [1, 3, 4, 18]) demonstrate that the term denotes different entities – subsystems, packages of software, binary code, etc. Therefore, the classification and ordering of components is an essential part of their definition. Because of this, in defining the component, we are generally following the point of view presented in [17] and [12].

First of all, the general component concept should be defined to unambiguously separate component from any constituent part in mereological sense, that is from any portion of a given entity. This clear distinction should be made to show the characteristic features of a component-based paradigm. So, any component is an autonomous unit, which provides services through contractually specified interfaces that hide interior of component from the outside view. To be more detailed, any component is a unit of composition, handlers resources, and has its life cycle.

First of all, components are classified as primitive (atomic) and composed components, which consist of primitive and/or compound ones. Interface-based composition is necessary to provide a new service using the existing components.

All the listed above fundamental features of component are to be used for defining components at lower levels, i. e. the components should be ordered. This ordering relation is transitive and asymmetrical. In other words, component of a specific kind inherits features of the most general one. In the enterprise system engineering context there are three basic kinds of components: business components, information system components, and software components (Figure 2).

Business components make up the basis for business system development, information system components – for information processing development, and software system components – for software systems development. Business components are business service providers and add value meaningful to an enterprise system [12]. They require services of each other, as well as (but not necessary) of information system components. Therefore, compound business component as a rule consists of business and information system components. Business components encapsulate business entities: processes, functional entities, resources, etc.



**Figure 2.** Taxonomy of components of enterprise system

Information system components provide information processing services. As information processing is a supporting process [8], IEIS components do not add value for external and internal users of an enterprise system. However, IEIS components support business level activity and are meaningful to an enterprise system. In collaboration with business components they can reshape business level services, increase their effectiveness. IEIS components require services of each other, as well as (but not necessary) of software system components. These components encapsulate IEIS entities such as information processing processes, functional entities, information objects, etc.

IEIS builders need to determine what, if any, of information processing should be computerized and how software systems can support this processing. Software system components are responsible for automatization of processes or/and their particular parts. These components encapsulate data, applications, digital devices, etc., and require services of each other.

The development of components (including the development for reuse and the development with reuse) can be quite complicated process; components as a rule are long-lived entities. Consequently, it is important to define the activities to be performed and work products to be produced, or, in other words, component has its life cycle model, which is a constituent part of a definition of component [19]. A life cycle concerns also the implementation issues of nonfunctional requirements, technological aspects. That is, life cycle model includes references to a component implementation model, component documentation model, and maybe to other models.

## 2.2. The Component-Based Enterprise Systems – Myth or Reality

IEIS are part of any business system and in their turn are enterprises: IS organizational structures come in all shapes and sizes, IS processes include maintenance, support, development, outsourcing, anticipating technologies, etc. Previous sections have assumed that IEIS (as well as enterprises) can be considered and designed as compositions of components. However, does the component-oriented development make sense and to what extent can it be realized?

There are two kinds of enterprises: functionally oriented (classical) and process-oriented [20]. The classification is based on one criterion – how the work is organized. It is important to emphasize that this grouping of enterprises reflects their generic architecture, includes their technical (views of economic theories) and behavioral aspects (views of behavioral theories).

Classical enterprises organize their work around the highly specialized tasks. Work is broken down into its simplest tiny tasks. The lines of authority and reporting are clearly drawn, “people involved in a process look *inward* toward their department and *upward* toward their boss” [20]. We can characterize functionally oriented enterprises in terms of component-based paradigm as follows:

- have small granularity elements,
- have tightly related parts,
- provide lower level services,
- have limited forms of connectors (because of accountability through the rigid chain of command),
- are inflexible.

Process-oriented enterprises reunify the specialized tasks into coherent business processes. These enterprises have clearly defined sets of activities for all its transactions, and a set of activities, taken together, produce a result of value to a user. In other words, enterprises are focused to provide services to their customers. The process-oriented enterprises can be characterized as follows:

- higher-order services (composition of specialized tasks is a result of value to a user),
- loosely coupled constituent parts of an enterprise (because of the absence of rigid vertical structures),
- large and small granularity of these constituents,
- are interface-based (because of the contracts between the system and customers to provide services),
- support of multiple interfaces and of different forms of dynamic binding (because of persistently changing environment).

These characteristics can be thought of as preconditions, which have to be verified for component-based paradigm to be applied. Thus, we argue that process-oriented enterprises and consequently their IEIS meet preconditions to component-based development.

### 2.3. Component as Abstraction in IEIS Development

Integrated enterprise information systems development is based on a number of engineering principles, i. e. on general methodological premises. These general principles include, but are not limited to: decomposition, abstraction, unification, and openness. These four principles represent what we believe to be the most important requirements.

The term abstraction has two meanings in this paper. First, component abstraction is thought of as general methodological principle, i. e. concept component is regarded as abstract construct, which should be used in the all stages of component-oriented IEIS development and may be materialized in different ways. Second, abstraction is thought of as a technique that allows one to focus on a portion of system, project, process, etc. while ignoring other features of these entities.

IEIS are very sophisticated systems, therefore the reuse and complexity management are critical problems facing their developers. Component-based development is a viable alternative to other approaches in this context, so we will discuss the concretization of the above listed engineering principles for component-based IEIS development.

The enterprise engineering artifacts (objects, processes, tasks, etc.) should be resolved into constituent parts: compounds and/or atomic elements. A decomposition principle enables us to reduce the complexity of IEIS and its development process. Functional decomposition, data decomposition, structural decomposition, object decomposition, task-oriented decomposition, etc. are well known decomposition techniques. A decomposition principle is characterized by a type of constituents (modules, functional units, etc.) and decomposition technique. However, the question “what decomposition should be used in component-based IEIS development?” is still open. It is obvious that neither functional, nor task-oriented decomposition is sufficient.

Abstraction principle in enterprise engineering implies three aspects:

- information hiding,
- iterativeness, and
- reusability.

Information hiding means that constituent parts of an entity include only the relevant information at each abstraction level. It helps the developers to manage complexity in large enterprise systems focusing on the few details at a time. On the other hand, information hiding supports a separation of concerns what is essential in large-scale systems development. It is a technique of breaking an artifact into distinct pieces of interest. For example, in programming abstraction means the distinction between what software unit does and ignores how it does it. The different abstraction levels are associated with iterations, i. e. any enterprise engineering asset is a refined outcome of an iterative development process. This fact highlights the problem of proper decomposition method, because of the following:

- all constituent parts of an entity at some abstraction level must be independent of each other – it should be possible to separately develop any constituent part;
- a set of constituent parts of an entity must be complete – it should be possible to develop all the required artifacts by combining the constituent parts into meaningful whole.

Reusability is the third value of abstraction in systems development. The identical artifacts at a higher abstraction level can be quite different at a lower one. It means that there is a one-to-many mapping between an abstract artifact and its more detailed counterpart. So, the abstract enterprise engineering assets can be reused in many different domains. Abstraction enlarges the scope of usability. For example, in software engineering, procedural abstraction includes a parameterization mechanism for sharing code. One can change the value of the parameters to reuse the procedure in different domains.

The unification principle requires all the system’s constituent parts to be consistent, i. e., conforming to the same principles, generally accepted rules, standards. Unification enables the developers to enhance reuse, to handle all kinds of change management problems more easily and flexibly. Enterprise system is three-layered system, and systems of all these layers should be defined in unified manner [8]. The same should be ensured in any layer and within the systems of the layer. In other words, IEIS constituent parts should be unified.

Openness in enterprise engineering enables extendibility and operation in different environments; it includes the development and evolution of IEIS through cooperation with different vendors. This principle is a prerequisite for the connectivity and

interoperability of systems and their constituents. The openness should be ensured in any layer and within the systems of the layer.

A paradigm is the set of common beliefs and agreements shared among scientists and practitioners about how the development problems should be understood and addressed [21]. It includes methodological premises, general principles, so the most general principles of systems development should be concretized to define a certain paradigm. (Let us remember, that procedure calls were the first abstraction realization mechanisms provided for programming.) Now we are in the position to specify the realization of general principles of component-based IEIS development paradigm. We argue that components should stand for constituent parts to realize the decomposition principle, and component abstraction should be used to realize the abstraction principle in component-oriented IEIS development.

First, it is possible to design entities that abstract over any syntactic category of modeling language, provided only that the categories ensure an appropriate correspondence between relevant elements in the world and model. We stand for component abstraction to realize the abstraction principle for the following reasons:

- Components are meaningful at all three layers of enterprise system. Components are consistent pieces of analysis, design, and implementation. Thus, this abstraction forms the basis of unified enterprise modeling environment. The need for such environment was highlighted in [8].
- Components provide a higher level of abstraction [22].
- Component abstraction has the secondary effect; it ensures perhaps the most important principle in IEIS development – the openness of system.

Second, component abstraction does not overload a set of abstractions in information systems development paradigm. At first glance, packages appear to be techniques for abstraction and information hiding. However, unlike component, package is a general-purpose mechanism for organizing elements into groups. Components, packages, partitions, models [23] and other modularization techniques have different semantics and purpose. Thus, a proper set of realization techniques to modularize entities should be defined in IEIS development methodology, as well as properly used in different IEIS development stages.

### **3. IEIS Components**

A number of components operate at the information technology level. Any component of this level is a constituent part of particular information system (sometimes it can be shared by several IS) and is used to implement, at least partly, a particular IEIS component. A number information systems components operate at the information processing level. Any IS component supports a particular business system but sometimes it can be shared by several business systems. Thus, not only all the components at all the levels of an enterprise should be composed properly, but also proper relations between different level components should be defined.

In IEIS the emphasis is shifted from selecting individual components to selecting sets of components that work together. Moreover, these sets consist of different types of components; for example, software and hardware components. So, it is necessary that systems operating at enterprise system layers were conceptualized in uniform manner. In other words, systems at all three levels should be perceived using the

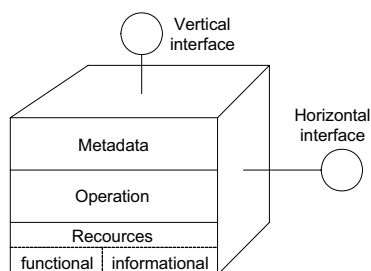
system of compatible categories. Thus, a set of IEIS components and their types is always linked to ontological assumptions about the enterprise system. In this paper, the ontological model of IEIS is based on M. Bunge [24] ontology, which is thought of as top-level ontology [25]. The Enterprise Ontology [26] is used as lower-level system ontology [9] to define basic IEIS modeling concepts. More exactly, an IEIS model is thought of as a set of activities, resources, goals, organization, products, and services. The rest of this section addresses the structural aspects, granularity and types of IEIS components.

### 3.1. Structural Constituents of IEIS Component

IEIS component first of all is a component, i. e. an autonomous entity that has defined interface (or interfaces) and can be plugged into a system. To define the constituents of IEIS component we consider an abstract operational system as the upper-level system and think of the IEIS components as a kind of this system. Operational system is a set of interrelated processes performed by certain processor (or by a network of processors) to achieve certain goals while producing particular outcome [8]. So, the IEIS components are characterized by *process* and *processor* models, as well as *resource* model and *service/product* model. We can clarify now the basic structural constituents of IEIS component.

The entities within the information system such as people, calculators, computers, documents, etc. are used to produce information processing services. All these entities are resources, which we categorize into two main parts: functional and informational resources (Figure 3). In other words, resources are required to run the processes and are manipulated (used, produced, modified, etc.) through these processes. So, processors are functional resources, and in case of computerized information system are, for example, hardware and employees of IS department. Informational resource is surrogate (in other words, representation) of any thing of business that is of interest to information processing processes. Informational resource holds data or knowledge about business entities and other entities of information system (there are entities in IEIS that are not present in business system) or is a record of socially constructed reality [27]. Informational resource is not obligatory constituent. Components (so called services) can operate on the external informational resources.

Operation encapsulates processes or series of actions (such as subtract, multiply, or shift, copy) that certain processor performs, and is thought of as functional aspect of IEIS component. This constituent of IEIS focuses on the activities performed by the



**Figure 3.** Constituents of IEIS component

information system, produces the required effects and changes the state of informational resources.

Metadata is supporting constituent and is used to gain insight into a component. Metadata contain all the information for use and reuse. Component as autonomous unit is uniquely identified; it is of different types, versions, and of different development states. Metadata contain or refer to full documentation, allows tools to know how to use, configure, and run them; can include the pointer to the ontology used to ensure the adequate understanding and interpreting of the provided services.

A service/product model of IEIS component shapes the interfaces, which present a definition of work for others that can be offered by this component. Thus, IEIS components interact through their interfaces. Two types of interfaces (Figure 3) can be distinguished: horizontal and vertical (the terms are borrowed from [21]). The horizontal interfaces are necessary to ensure relationships with the other components of the same level and type, for example, application programs are composed of software components. The vertical interfaces are necessary to define interactions between the components of different levels and types, for example, between the computer platform and software components, between the data and application program components.

In summary, we can define four essential structural constituents of IEIS components. These constituents (except metadata) in their turn are components or their compositions.

### 3.2. Primitive and Compound IEIS Components

The structuring of enterprise information systems traditionally is based on the functional areas and the support provided by the system [28, 29]. This classification focuses on functionality and the domain of interest in which certain system is used. However, it does not answer to the main question – what are the characteristic features of information system which distinguish it from the other types of systems. On the other hand, systems are usually developed by including proper existing constituents, reusing proper architectures and frameworks ([1] – is the example of software systems development framework). In other words, knowing the type of system guides the development process. This implies the definition and use of so called system level IEIS components, i. e. parts that are constituents of any information system. The following are examples of such IEIS components: data entities, documents and their bases, registers, libraries, help desks, messaging systems, corporation memories, and security systems.

IEIS component we define as the result of construction process – by the following tuple:

$$ISC = \langle ISP, p^{(k)}, \omega^{(l)}, Cr \rangle,$$

where

$ISP = (ISP_1, ISP_2, \dots, ISP_m)$  – a set of primitive IEIS components;

$p^{(k)}$  – a set of permissible primitive constructors;

$\omega^{(l)}$  – a set of permissible higher-level constructors;

$Cr$  – constraints.

The elements of sets  $ISP$  are primitive structural constituents of any compound IEIS component.  $\tau(isp_{j'} \in ISP_j) \neq \tau(isp_{j''} \in ISP_{j+1})$ , where  $\tau$  is a function defined over components, and returns type of component. Functional resource, informational resource, computational activity, service activity, and informational activity are the basic types of IEIS components (see Section 3.1).



Primitive constructors ( $p^1, \dots, p^k$ ) are two-placed, and these constructors are used to define binary relations over a set of IEIS components. For example, IEIS component uses the service of another one. Any set  $ISP_i$  has its associated set  $(p^1, \dots, p^n)$ ,  $k \geq n$ . Composition using primitive constructors is horizontal interfaces based, and it results in compound component of the same type as its constituents. Higher-level constructors ( $\omega^1, \dots, \omega^l$ ) are many-placed and underlie the development of new types of IEIS components. Composition using higher-level constructors is vertical and horizontal interfaces based.

Constraints are put on the system of components as a whole, on the set of permissible types of components, on the set of primitive IEIS components, on the permissible combinations of components to assembly IEIS component.

#### 4. Related Works

Specification of aspects and concerns of complex enterprise systems in consistent component-oriented manner is presented by Z. Stojanovic and A. Dahanayake [12]. They define different types of components through different enterprise system models. Authors propose three component stereotypes – business component, system component, and distribution component. Business components “define business services for which automated system support is required” [12]. System components provide lower grained services and have more technical meaning. Distributed components are system components in distribution context, i. e., system components are used as an entry for the distributed system design.

Business Component Factory [3] – component-based development for an enterprise – introduces 3 levels of component granularity: distributed component, business component, and business component system. However, all three are considered as different kinds of software components. Business component implements autonomous business entity, and group of business components cooperate to deliver the required functionality for business needs.

In summary, our main claim regards the clear distinction between information system components and supporting software components. In other words, development of information system should not be considered in the context of automation of business processes. Development of IS and its components in the first stages of life cycle is technology independent, i. e. component is an abstraction, which is supported by development infrastructure. In other words, IEIS development should be thought of as a separate stage in enterprise system development process.

T. Le Dinh [13] aims at supporting the development of component-based information systems. He defines architecture of each enterprise as consisting of business and informatics (information systems) layers separated by intermediate information layer. The information layer is necessity to deal with “conceptual specifications for effective IS development” [13]. IS component is an autonomous artifact of IS that can be developed independently from the other IS components and includes static, dynamic, and integrity rules. However, author’s research interests address the problem of managing and coordinating information resources used in IS development.

A short note on the basic systems engineering principles. Catalysis approach [1] for the systematic business-driven development of component-based software systems is founded on three principles: abstraction, precision, and pluggable parts. However,

abstraction principle in this approach highlights the information hiding. "To abstract means to describe only those issues that are important for a purpose, deferring details that are not relevant" [1]. Here we would like to notice that important abstractions include requirements, architecture, business rules and other aspects of IS development. In addition, implementation of abstraction principle depends upon the constructs of a modeling language, namely upon UML in Catalysis approach.

## 5. Concluding Remarks

The state of affairs in the component-based development of integrated enterprises information systems can be characterized as follows: component-oriented approaches in this field are at the very beginning. Business entities and processes are implemented as software components, i. e. main focus is on the reuse of information technology level software artifacts. On the other hand, component-based paradigm is becoming more and more mature and can be used for enterprise information systems development. In order to do this we conceptualize business, information, and software systems in uniform component-oriented manner. IEIS components support business level activity and encapsulate IEIS entities. The discussed component-based IEIS development paradigm provides the notion of IEIS component and realization of the most general systems development principles. The consequence of uniform conceptualization - IEIS components are characterized by process and processor models, as well as resource model and service/product model. The proposed IEIS component definition emphasize different types of components and the necessity of means to assembly these components of different types into the compound ones.

This paper lays the ground position for additional research. The work needs to progress along a number of lines to fully enable component-oriented IEIS development. The first steps should be to propose realization technique of general decomposition principle, to define and specify the primitive IEIS components and constructors.

## References

- [1] D. F. D'Souza and A. C. Wills, *Objects, Components, and Frameworks. The Catalysis Approach*. Addison-Wesley, 1999.
- [2] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, 1998.
- [3] P. Herzum and O. Sims, *Business Component Factory: a Comprehensive Overview of Business Component Development for the Enterprise*. John Wiley & Sons, 2000.
- [4] A. Ju An Wang and K. Qian, *Component-Oriented Programming*. Wiley-Interscience, 2005.
- [5] F. Bachmann et al. *Volume II: Technical Concepts of Component-Based Software Engineering*, Technical Report CMU/SEI-2000-TR-008 ESC-TR-2000-007, Software Engineering Institute, 2000.
- [6] J. Cheesman and J. Daniels, *UML Components: a Simple process for Specifying Component-Based Software*. Addison-Wesley, 2001.
- [7] Z. Stojanovic, *A Method for Component-Based and Service-Oriented Software Systems Engineering*. Delft University of Technology, The Netherlands, Doctoral Dissertation, 2005.
- [8] A. Caplinskas, A. Lupeikiene, and O. Vasilecas, Shared conceptualisation of business systems, information systems and supporting software. In: *Databases and Information Systems II*, H.-M. Haav and A. Kalja, Eds., Kluwer Academic Publishers, 2002, pp. 109-320.
- [9] A. Caplinskas, A. Lupeikiene, and O. Vasilecas, The role of ontologies in reusing domain and enterprise engineering assets. *Informatica*, vol. 14, 2003, pp. 455-470.
- [10] A. Caplinskas, An ontology-based approach to enterprise engineering. *Computer Science Reports*, vol. 14/03, BTU Cottbus, 2003, pp. 22-30.

- [11] R.J. Wieringa, H. M. Blanken, M. M. Fokkinga, and P. W. P. J. Grefen, Aligning application architecture to the business context. In: *Advanced Information Systems Engineering*, J. Eder and M. Missikoff, Eds., LNCS 2681, Springer, 2003, pp. 209–225.
- [12] Z. Stojanovic and A. Dahanayake, Components and viewpoints as integrated separations of concerns in system designing. In: *Proc. of the Workshop on Aspect-Oriented Design* (in conjunction with the 1st International Conference on Aspect-Oriented Software Development), 2002, pp. 22–26.
- [13] T. Le Dinh, *Information System Upon Information Systems: a Conceptual Framework*. University of Geneva, Doctoral Dissertation, 2005.
- [14] Y. F. Jarrar, A. Al-Mudimigh, M. Zairi, ERP implementation critical success factors - the role and impact of business process management. In: *Proceedings of the IEEE International Conference on Management of Innovation and Technology ICMIT 2000*, Singapore, 12–15 November, 2000, pp. 122–127.
- [15] S. Sadagopan, Enterprise Resource Planning. In: *Encyclopedia of Information Systems*, H. Bidgoli, Ed., Elsevier Science, 2003, pp. 169–184.
- [16] M. P. Sena, Enterprise computing. In: *Encyclopedia of Information Systems*, H. Bidgoli, Ed., Elsevier Science, 2003, pp. 157–167.
- [17] M. Volter, A taxonomy for components. *Journal of Object Technology*, vol. 2, July–August 2003, pp. 119–125.
- [18] P. Kruchten, *The Rational Unified Process: an Introduction*. Addison-Wesley, 2003.
- [19] A. Lupeikiene and V. Giedrimas, Component model and its formalisation for structural synthesis. *Scientific Proc. of Riga Technical University, Computer Science*, vol. 22, 2005, pp. 169–180.
- [20] M. Hammer, J. Champy, *Reengineering the Corporation*. Nicholas Brealey Publishing, 1996.
- [21] T. Kuhn, *The Structure of Scientific Revolutions*. University of Chicaho Press, 1962.
- [22] H. Mili, A. Mili, S. Yacoub, *Reuse-Based Software Engineering: Techniques, Organization, and Control*. John Wiley & Sons, 2002.
- [23] Unified Modeling Language: Superstructure. Version 2.0. Object Management Group, Inc., (2005, Aug.) [Online]. Available: <http://www.omg.org/technology/documents/formal/uml.htm>.
- [24] M. Bunge, *Ontology I: The Furniture of the World, Treatise on Basic Philosophy*, vol. 3, D. Reidel Publishing Company, 1977.
- [25] N. Guarino, Formal ontology and information systems. In: N. Guarino (ed.), *Formal Ontology in Information Systems*. Proceedings of FOIS'98, Trento, 6–8 June 1998. Amsterdam, IOS Press, pp. 3–15.
- [26] M. Uschold, M. King, S. Moralee, and Y. Zorgios, The Enterprise Ontology. *The Knowledge Engineering Review*, Special Issue on Putting Ontologies to Use, vol. 13, pp. 31–89, 1998.
- [27] R. M. Colomb, R. Weber, Completeness and quality of an ontology for an information system. In: N. Guarino (ed.), *Formal Ontology in Information Systems*. Proceedings of FOIS'98, Trento, Italy, 6–8 June 1998. Amsterdam, IOS Press, pp. 207–217.
- [28] S. Alter, *Information Systems: a Management Perspective*. The Benjamin/ Cummings Publishing Company, 1996.
- [29] K. C. Laudon, J. Laudon, *Essentials of Management Information Systems: Organization and Technology*. Prentice Hall, 1996.

This page intentionally left blank

# Web Engineering

This page intentionally left blank

# A Web Service-Oriented Architecture for Implementing Web Information Systems

Flavius FRASINCAR <sup>a,b</sup>, Geert-Jan HOUBEN <sup>a,c</sup>, and Peter BARNA <sup>a</sup>

<sup>a</sup> *Eindhoven University of Technology*

*Den Dolech 2, 5612 AZ Eindhoven, the Netherlands*

<sup>b</sup> *Erasmus University Rotterdam*

*Burgemeester Oudlaan 50, 3062 PA Rotterdam, the Netherlands*

<sup>c</sup> *Vrije Universiteit Brussel*

*Pleinlaan 2, B-1050 Brussels, Belgium*

**Abstract.** Web services are one of the most popular ways for building distributed Web Information Systems (WIS). In this paper we propose a distributed implementation of the Hera methodology, a methodology based on models that specify the different aspects of a WIS. The Web services used in the implementation are responsible for capturing the required data transformations built around specific Hera models. A service orchestrator coordinates the different Web services so that the required WIS presentation is built. Based on the degree of support of the user interaction with the system two architectures are identified: one for the construction of static applications, and another one for the building of dynamic applications.

**Keywords.** Web Information Systems, Web services, RDF(S)

## Introduction

The Web The Web has more than three billion pages and around half a billion users. Its success goes beyond national frontiers and imposes it as the first truly global information medium. While the nineteenth century was dominated by the "industrial revolution", the beginning of this new century is marked by an "information revolution" with the Web as its main "engine".

A Web Information System (WIS) [1] is an information system that uses the Web paradigm to present data to its users. In order to meet the complex requirements of a Web Information System several methodologies have been proposed for WIS design. In the plethora of proposed methodologies we distinguish the model-driven ones that use models to specify the different aspects involved in the WIS design. The advantages of such model-based approaches are countless: better understanding of the system by the different stakeholders, support for reuse of previously defined models, checking validity/consistency between different design artifacts, (semi-)automatic model-driven generation of the presentation, better maintainability, etc.

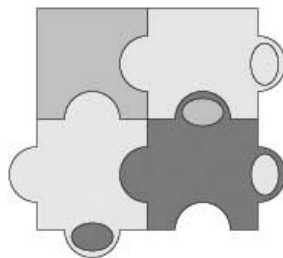
Based on the principle of separation of concerns, model-driven methodologies like OOHDM [2], WebML [3], UWE [4], SiteLang [5], and Hera [6] distinguish: a conceptual model that describes the schema of the (multimedia) data to be presented, a navi-

gation model that specifies how the user will be able to navigate through the data, and a presentation model that gives the layout and the low-level characteristics (e.g., font size, color, etc.) of the hypermedia presentation to be generated. For the model representations, these methodologies make use of different technologies: OOHDM is based on the OO paradigm, WebML offers a set of visual units (serialized in XML) to be graphically composed, UWE is based on the UML notation extended with a Web UML profile, SiteLang suggests a mathematical language (story algebra), and Hera uses Semantic Web technology to make explicit the semantics captured in a certain model and to subsequently exploit this semantics.

Today there is an increasing need for making WIS components interoperable. An isolated WIS is not able to provide all the information/computational power required by an organization. Web services (WS) appear to be the most popular mechanism to support application interoperability. Their success is based on the fact that they are cross-platform and cross-language distributed applications. The fact that the provider and the requester of a WS are loosely coupled ensures application robustness. A disadvantage of such an approach is the XML messaging involved, which affects application speed. Nevertheless using appropriate optimization techniques and/or fast computers/networks this disadvantage can be overcome. We distinguish between two classes of WS: WS that provide data-on-demand, and WS that offer computation-on-demand. As an example for the first type of WS we mention WS offered by Amazon or Google, and for the second type of WS we refer to the services developed in the Grid project [7].

Different science communities saw the opportunity of using WS to enable data sharing between their systems. For example, the geoscience community came up with a service-oriented solution for the interoperability of geographical information systems aiming at providing better forecasts. The proposed WS solution [8] is compared with a puzzle (of WS) as depicted in Figure 1. In this puzzle, shapes represent interfaces, and colors (shades of gray for a black and white printing) stand for data models/encodings. Interfaces are the operations that a WS provides, and data models/encodings are the input/output parameter types of the WS operations.

Most model-driven methodologies for WIS design have modular implementations made from components driven by different models. Based on the Hera methodology, this paper proposes the use of WS in realizing the different components and aggregating them in a WIS. In this way the software plug-and-play vision can be realized in the context of WIS.



**Figure 1.** Web services puzzle.



## 1. Hera Methodology

Hera [9] is a model-driven methodology that uses Semantic Web technologies for building WIS. There are two main phases in Hera: the data collection phase responsible for integrating the data coming from different sources, and the presentation generation phase responsible for constructing a Web presentation of the integrated data. The focus of the paper is on the second phase of Hera, nevertheless some of the ideas presented here can be applied also for the data collection phase.

As a model-driven methodology, Hera distinguishes design models (specifications) that capture the different aspects of the presentation generation phase in a WIS. The conceptual model (CM) describes the schema of the data that needs to be presented. The application model (AM) expresses the navigational structure to be used while browsing through the data. The presentation model (PM) specifies the look-and-feel aspects (layout and style) of the presentation. The Hera methodology also proposes an implementation which builds a WIS based on the previously specified models.

One of the characteristics of the Hera methodology is its specific focus on personalizing the WIS based on the user characteristics and user browsing platform. The user profile (UP) is responsible to store the attribute-value pairs that characterize the user preferences and platform properties. These properties are used to customize the presentation generation before the user starts the browsing session. The adaptation technique utilized for implementing WIS personalization is based on adding visibility conditions to elements appearing in the different Hera models. These conditions make use of the data stored in the UP in order to define the presence/absence of a model element.

Another feature that Hera supports for a WIS is the ability to consider the user interaction with the system (by user interaction we mean here the interaction by forms in addition to merely link following) before the next Web page is generated. For this purpose the User Session (US) was defined to store all the data created at run-time by the user. The forms and the queries responsible for building new data are described in the AM.

All Hera models are represented in RDF(S), the Semantic Web foundation language. In this way we were able to cope with the semi-structured aspects of data on the Web (by using the loose RDFS schema definitions), support application interoperability (by making available the data semantics as RDFS descriptions), reuse existing models (by utilizing the refinement mechanisms supported by RDFS), etc. RDF(S) also allows for compact representations of the models as intensional data can be provided at run-time by an inference engine. For the user profile we did reuse an existing RDFS vocabulary, User Agent Profile (UAProf) [10], an industry standard for modeling device capabilities and user preferences.

As a query language we used SPARQL [11] one of the state-of-the-art query languages for RDF models. Differently than other RDF query languages (e.g., RQL) SPARQL allows for the construction of new models in addition to extraction of relevant information from existing models. This feature was useful for generating new data used for populating the US.

The implementation of the Hera methodology is based on several data transformations operating on the Hera models. There are three types of transformations: application-dependent, data-dependent, and session-dependent. The application-dependent transformations operate on CM, AM, and PM, and do not consider the instances of these mod-

els (e.g., model adaptation). The data-dependent transformations convert one model instance into another model instance (e.g., CM instance into AM instance). The session-dependent transformations make use of the data provided by the user (e.g., populating the US with form-related data).

In order to support the presentation of the Hera models we use a running example, a WIS that depicts Vincent van Gogh paintings, and allows the users to order posters of the displayed paintings. Figure 2 shows a snapshot of the hypermedia presentation for the considered WIS. It presents a Self Portrait painting of Vincent van Gogh.

### 1.1. User Profile

The user profile (UP) specifies the user preferences and device capabilities. It is a CC/PP vocabulary [12] that defines properties for three components: *HardwarePlatform*, *SoftwarePlatform*, and *UserPreferences*. Figure 3 illustrates an excerpt from a UP. For the hardware characteristics one such property is *imageCapable*. This property is set to *Yes* for devices that are able to display images. The software characteristics have, among others, the property *emailCapable*. This property is set to *Yes* for browsers that are able to provide email functionality. For user preference characteristics one such property is *isExpert*. This property is set to *Yes* if the user is an expert of the application domain.

### 1.2. Conceptual Model

The conceptual model (CM) represents the schema of the data that needs to be presented. It is composed of concepts, concept relationships, and media types. There are two kinds of concept relationships: attributes which relate a concept to a particular media type, and inter-concept relationships to express relations between concepts. A CM



Figure 2. Presentation in Internet Explorer.

```

<ccpp:component rdf:ID="comp_ID1">
  <HardwarePlatform>
    <imageCapable>Yes</prf:imageCapable>
    ...
  </HardwarePlatform>
</ccpp:component>
<ccpp:component rdf:ID="comp_ID2">
  <SoftwarePlatform>
    <emailCapable>Yes</prf:emailCapable>
    ...
  </SoftwarePlatform>
</ccpp:component>
<ccpp:component rdf:ID="comp_ID3">
  <UserPreferences>
    <isExpert>Yes</isExpert>
    <levelOfVision>Normal</levelOfVision>
    ...
  </UserPreferences>
</ccpp:component>

```

**Figure 3.** Excerpt from UP serialization in RDF/XML.

instance (CMI) gives the data that needs to be presented. This data might come from an integration/retrieval service that is out of the scope of this paper. Figure 4 illustrates an excerpt from a CMI. It describes a Self Portrait painting of Vincent van Gogh. Such a painting exemplifies the Impressionism painting technique identified by Technique\_ID2. In the description one can find the year in which the painting was made, i.e., 1887, and the URL of the image depicting it.

The adaptation condition used in this case specifies that the image of the painting is shown only if the device is image capable.

```

<Painting rdf:ID="Painting_ID6">
  <name>
    <type:String>
      <type:data>Self Portrait</type:data>
    </type:String>
  </name>
  <year>
    <type:Integer>
      <type:data>1887</type:data>
    </type:Integer>
  </year>
  <picture>
    <type:Image rdf:about="http://.../SK-A-3262.ORG.jpg"
      a:condition="up:imageCapable = 'Yes'"/>
  </picture>
  <exemplifies rdf:resource="#Technique_ID2"/>
  <painting_by rdf:resource="#Painter_ID2"/>
</Painting>

```

**Figure 4.** Excerpt from CMI serialization in RDF/XML.

### 1.3. Application Model

The application model (AM) gives an abstract view of the hypermedia presentation that needs to be generated over the CM data. It is composed of slice and slice relationships. A slice is a meaningful presentation unit that groups media items coming from possibly different CM concepts. The AM is in this way built on top of the CM. A slice that corresponds to a page from the presentation is called a top-level slice. There are two kinds of slice relationships: aggregation which enables the embedding of a slice in another slice, and navigation which specifies how a user can navigate between slices. An AM instance (AMI) populates an AM with data from a CMI. Figure 5 depicts an excerpt from an AMI. It presents the slice `Slice.painting.main_ID6`, a top-level slice associated to the painting presented in the above subsection. This slice refers using slice aggregation relationships to four other slices. Each of the four slices contains a media item related to different painting/painter attributes.

The adaptation condition used in this example specifies that the year in which a painting was made is presented only for expert users. Other users, like for example the novice users, will not be able to view this information.

Let's consider now that the application is dynamic in the sense that the user is allowed to buy posters depicting paintings. In this case the user can place orders based on the currently displayed painting and a specified quantity (by means of a form) of this poster. The US stores the data that is created by the user at run-time. In our example the US contains the order to be added to the shopping trolley. Figure 6 shows an example of a query used in the AM to create an order (variable `x`) that stores the desired painting (variable `id`, the currently displayed painting), and the required quantity (variable `v`, the number input by the user in a form).

```
<Slice.painting.main rdf:ID="Slice.painting.main_ID6">
  <slice-ref rdf:resource="#Slice.painting.name_ID6"/>
  <slice-ref rdf:resource="#Slice.painting.year_ID6"/>
  <slice-ref rdf:resource="#Slice.painting.picture_ID6"/>
  <slice-ref rdf:resource="#Slice.painter.name_ID2"/>
</Slice.painting.main>

<Slice.painting.name rdf:ID="Slice.painting.name_ID6">
  <media>
    <type:String>
      <type:data>Self Portrait</type:data>
    </type:String>
  </media>
</Slice.painting.name>
...

<Slice.painting.year rdf:ID="Slice.painting.year_ID6"
  a:condition="up:isExpert = 'Yes'">
  <media>
    <type:Integer>
      <type:data>1887</type:data>
    </type:Integer>
  </media>
</Slice.painter.name>
```

**Figure 5.** Excerpt from AMI serialization in RDF/XML.

```

CONSTRUCT { ?x <rdf:type> us:Order .
             ?x <rdf:contains> ?y .
             ?y <rdf:ID> ?id .
             ?x <us:quantity> ?v }
WHERE       { BuyForm <bf:quantity> ?v .
             Session <var:paintingid> ?id }

```

**Figure 6.** A SPARQL query.

The SPARQL language was extended with new constructs like: URI generator (e.g., the order identifier), aggregation functions (e.g., counting the number of orders that are in the trolley), and delete statement (e.g., the user should be able to delete orders from the trolley).

#### 1.4. Presentation Model

The presentation model (PM) defines the look-and-feel of the application. It is composed of regions and region relationships. A region is an abstraction for a rectangular part of the user display area. Each region corresponds to a slice that will provide the content to be displayed inside the region. A top-level region is a region that is associated to a top-level slice. There are two types of region relationships: aggregation relationships, which allow the embedding of a region into another region, and navigation relationships, which allow the navigation from one region to another region.

A region has a particular layout manager [13] and style associated with it. There are four layout managers: *BoxLayout*, *TableLayout*, *FlowLayout*, and *TimeLayout*. These layout managers describe the spatial/temporal arrangement of the regions in their container region. They allow us to abstract from the client-dependent features of the browser's display. The style specifies the fonts, colors, backgrounds, etc., to be used by a certain region. The definition of the style properties allows a uniform representation of style information irrespective of the CSS compatibility of the client. Regions that do not have a style defined inherit the style of the container region. The top-level regions always need to have a style defined.

A PM instance (PMI) populates a PM with data from an AMI. Figure 7 depicts an excerpt from a PMI. It presents the region *Region.painting.main\_ID6*, a top-level region associated to the considered Self Portrait painting. This region refers using region aggregation relationships to four other regions. Each of the four regions contains a media item related to different painting/painter attributes. The top-level region has a *BoxLayout* associated with a *y* orientation and the style uses *times* fonts.

The adaptation condition presented in this example states that a default style with normal fonts should be used for users with a normal level of vision. For users with a poor level of vision a default style with large fonts is prescribed.

## 2. Hera Web Service-Oriented Architecture

The previous implementation of Hera was based on one centralized application. This application is made of several components each responsible for performing a specific set of data transformations. In this paper we will show how one can distribute such an implementation by mapping components to WS. One of the advantages of using such an im-

```

<Region.painting.main rdf:ID="Region.painting.main_ID6"
    layout="#BoxLayout_ID1"
    style="#DefaultStyle_ID1"
    style="#DefaultStyle_ID2">
    <region-ref rdf:resource="#Region.painting.name_ID6"/>
    <region-ref rdf:resource="#Region.painting.year_ID6"/>
    <region-ref rdf:resource="#Region.painting.picture_ID6"/>
    <region-ref rdf:resource="#Region.painter.name_ID2"/>
</Slice.painting.main>

<BoxLayout rdf:ID="BoxLayout_ID1"
    axis="y"
    width="100%"
    ...
</BoxLayout>

<Style rdf:ID="DefaultStyle_ID1"
    a:condition="up:levelOfVision = 'Normal'"
    font-family="times"
    font-size="normal"
    ...
</Style>

<Style rdf:ID="DefaultStyle_ID2"
    a:condition="up:levelOfVision = 'Poor'"
    font-family="times"
    font-size="large"
    ...
</Style>

```

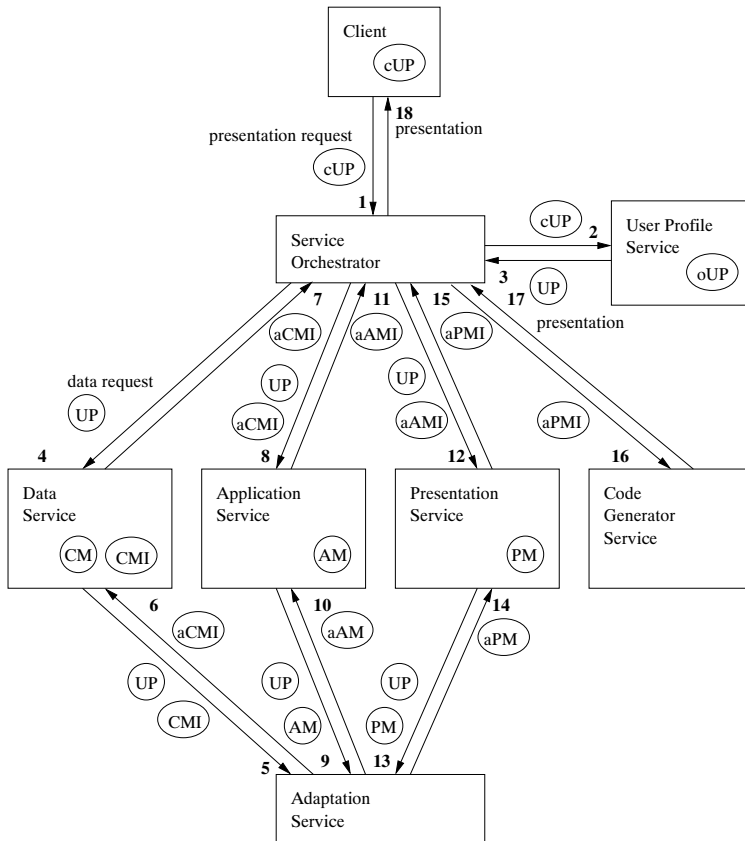
**Figure 7.** Excerpt from PMI serialization in RDF/XML.

plementation is the loose coupling of Hera WS which can help realize the plug-an-play software vision in the context of WIS. For example a WIS can be constructing by composing a WS responsible for delivering the data, a WS that builds a navigations structure over this data, a WS that adds the layout and style details, and a WS that generates the presentation code.

The WS solution used for realizing the distributed Hera architecture has clear advantages compared to their predecessors CORBA, J2EE, and DCOM [14]. First of all WS are based on the XML document paradigm, a human readable language that abstracts from the implementation details. WS interfaces are specified in the universally accepted Web Service Definition Language (WSDL) [15], an XML-based language. Last but not least, WS use the popular HTTP protocol as the carrier of exchanged messages.

### 2.1. Hera Web Service-Oriented Architecture for Static Applications

Figure 8 presents a Web service-oriented architecture (WSOA) for the Hera implementation based on six WS: the Data Service, the Application Service, the Presentation Service, the Code Generator Service, the User Profile Service, and the Adaptation Service. Note that this architecture is used to build static applications, i.e., applications for which the user is unable to change the presentation to be generated. After the description of this architecture we present another architecture that is used to generate dynamic applications, i.e., applications that take in consideration the user input data (from forms) before generating the next page.



**Figure 8.** Web service-oriented architecture for static applications.

The User Profile Service is responsible for providing the UP to be used by the application. The Adaptation Service is able to execute the adaptation specifications for the CM, AM, and PM. The core services are: the Data Service, which provides the data, the Application Service, which builds the navigation structure over the data, the Presentation Service which sets the layout and style of the presentation, and the Code Generator, which constructs the actual presentation. We call these services core services because they can form a WIS even when the other services (i.e., the User Profile and the Adaptation Service) are absent.

In addition to these WS the application uses two other components: the Client which is the initiator of the presentation request, and the Service Orchestrator responsible for coordinating the collaboration among WS. The Service Orchestrator is the hub of the whole system as it is the interface with the client and the message dispatcher to the core WS. The communication between Client and services is done at SOAP [16] level which resides on top of HTTP while the communication between the Client (Web Server) and the Web Browser is done in plain HTTP.

Figure 8 also depicts the flow of information in our distributed system. Each message has a number associated that indicates the order in which these messages are exchanged.

The architecture components are depicted in rectangles, models are represented as ovals, and messages are shown as arrows. On the arrows one can read the models that are being passed, 'presentation request/data request' (as events), and 'presentation' which is the returned hypermedia presentation. All components are WS with the exception of the Client and Service Orchestrator. The models that are own by the application components are placed inside the corresponding rectangles.

First the Client sends a presentation request to the Service Orchestrator (step 1). It also passes the client UP (cUP), the UP residing on the Client (possibly given by the Web Browser). The Service Orchestrator sends the cUP to the User Profile Service (step 2). The User Profile Service computes an updated version of the UP, by integrating the cUP with the old User Profile (oUP), the user profile stored in this service. The profile attributes not defined in the cUP are taken from oUP, and the resulted UP replaces the oUP. The User Profile Service can be viewed as a shared memory service for user profiles. After the integration process the UP is passed back to the Service Orchestrator (step 3).

The Service Orchestrator sends a data request to the Data Service. It also passes the UP, the integrated user profile from the previous phase (step 4). The CM instance (CMI) and the UP are passed further to the Adaptation Service (step 5). The Adaptation Service has one task to fulfill: evaluate the conditions in the models (CMI, AM, PM) and prune the models from the elements that have an associated condition that does not hold. For the CMI these elements are concepts or media items. In our application the Adaptation Service works on an instance, i.e., the input data of the system (CMI), and two schema models (AM, PM), as the instances of the two schema models will be populated at run-time. The adapted CMI (aCMI) is passed back to the Data Service (step 6) which forwards it to the Service Orchestrator (step 7).

The Service Orchestrator sends the UP and the aCMI to the Application Service (step 8). The AM and the UP are forwarded to the Adaptation Service (step 9). The Adaptation Service removes the AM slices that have the associated condition not valid. The resulted adapted AM (aAM) is passed back to the Application Service (step 10). The Application Service transforms the aCMI into the aAMI, based on the specifications from the aAM. The resulted aAMI is forwarded to the Service Orchestrator (step 11).

The Service Orchestrator sends the UP and the aAMI to the Presentation Service (step 12). The PM and the UP are forwarded to the Adaptation Service (step 13). The Adaptation Service selects the appropriate layouts and styles for the PM by removing the PM elements that have the associated condition not valid. The resulted adapted PM (aPM) is passed back to the Application Service (step 14). The Presentation Service transforms the aAMI into the aPMI, based on the specifications from the aPM. The resulted aPMI is forwarded to the Service Orchestrator (step 15).

The Service Orchestrator sends the aPMI to the Code Generator Service (step 16). The Code Generator Service transforms the aPMI into code interpretable by the user's browser. At the moment the intelligence of this transformation is encapsulated in the service logic itself, but in the future we would like to have an explicit model that maps PM elements to the code specifics. We did experiment with several code generators: HTML, cHTML, HTML+TIME, WML, and SMIL. The resulted presentation is passed back to the Service Orchestrator (step 17), which forwards it to the Client (step 18).



Note that this is not the only way in which the information can flow through the system. For example the adaptations can be performed first, i.e., before transforming one model instance into another model instance.

## 2.2. *Hera Web Service-Oriented Architecture for Dynamic Applications*

Figure 9 presents a WSOA implementation of Hera for dynamic Web applications. Differently than before where a full presentation was generated now we build one page-at-a-time as the application needs to take in account the user's input after each page before generating the next page. For this purpose we have added an extra service, the User Session Service responsible for collecting the user session data. We also have modified the logic of the Application Service so that the session data is considered in the data transformation that it performs.

Instead of presenting the full flow of information for this distributed architecture, we emphasize the major differences with respect to the distributed architecture for static applications.

At the beginning the Client sends besides the page request and the cUP, also the session data, which contains the data that emerges from the interaction of the user with the system (e.g., the data resulted from a form filling by the user). The presentation request (a presentation being a set of linked Web pages) used in the previous architecture is now replaced by a page request. The communication between the Service Orchestrator and the User Profile Service goes the same as before (steps 2 and 3).

The Service Orchestrator sends the page request, session data, and UP to the Application Service (step 4). In the previous architecture the Service Orchestrator could directly communicate with the Data Service as the full presentation was built at once. Here it is important to know the context of the presentation (what is the page that needs to be generated next), and this information is in the Application Service. For this reason the Service Orchestrator communicates first with the Application Service and not the Data Service.

The communication between the Application Service and Data Service (steps 5 and 8), and between the Data Service and Adaptation Service (steps 6 and 7) is similar as the communication between the Service Orchestrator and Data Service, and between the Data Service and the Adaptation Service from the previous architecture. The only difference is that the aCMI now corresponds only to the data necessary for computing the next Web page (instead of the whole presentation).

The Application Service sends the session data to the Session Service (step 9). The Session Service transforms the session data into a US instance (USI). After that the Session Service sends the USI back to the Application Service (step 10). The communication between the Application Service and the Adaptation service is the same as before (steps 11 and 12). Based on the aAM, the Application Service transforms the aCMI and USI into aAMI. As the aCMI, the aAMI contains only the data that is relevant for the current page. The computed aAMI is passed back to the Service Orchestrator (step 13).

The communication between the Service Orchestrator and the Presentation Service (steps 14 and 17) and between the Presentation Service and the Adaptation Service (steps 15 and 16) is similar as before. Also the communication between the Service Orchestrator and the Code Generator (steps 18 and 19), and between the Service Orchestrator and the Client (step 20) is done in the similar way as in the previous architecture.

The only difference is that the aPMI corresponds only to one page and the presentation is being replaced by one page.

Note that this is not the only way in which the information can flow through the system. For example the communication with the User Profile and the Adaptation Service can be performed only at the beginning (before the user starts browsing the presentation or only during the first run through the system). This is because both the user profile and the adaptation specifications are static, i.e., they are not influenced by the user interaction with the system.

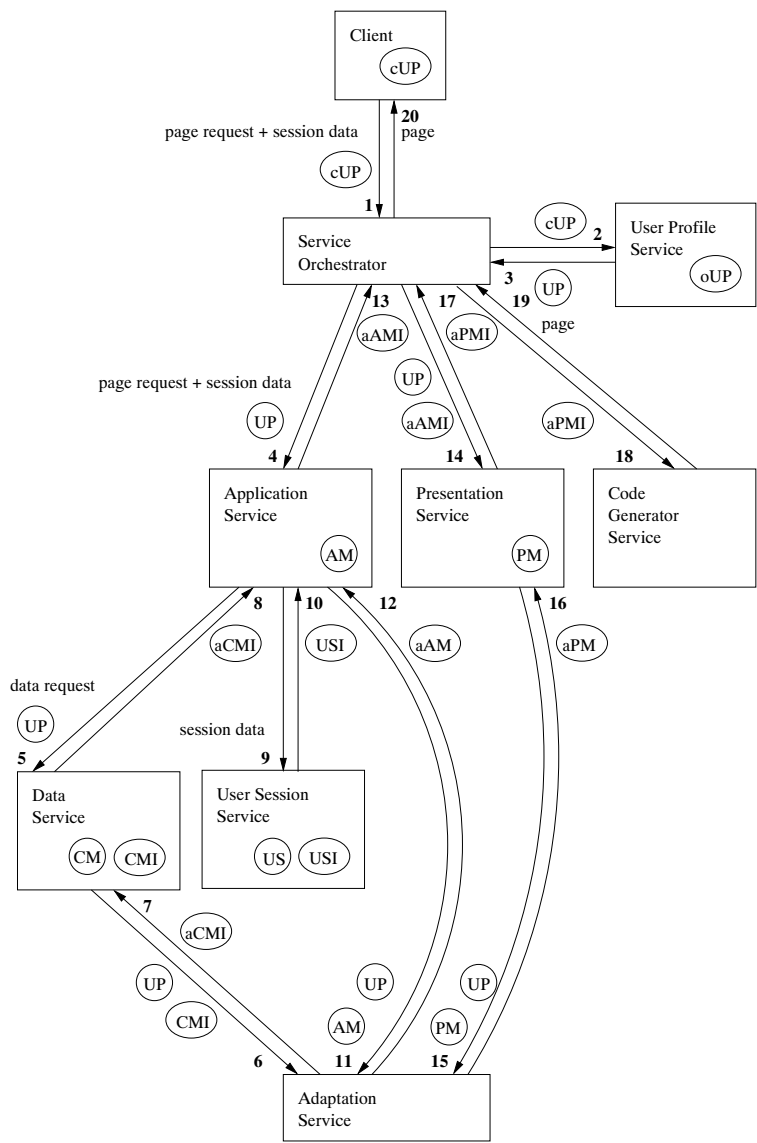


Figure 9. Web service-oriented architecture for dynamic applications.

### 2.3. Tools

In order to experiment with the proposed architecture a Java-based Hera tool was developed. Tomcat 4.1 [17] was used as the Web server that supports servlets. On this Web server we installed Axis 1.1 (Apache eXtensible Interaction System) [18], a SOAP 1.1 engine. By SOAP engine we mean a tool that supports both a SOAP server and SOAP clients. We did deploy on the SOAP server all WS. For their deployment we used appropriate Axis Web Service Deployment Descriptors. The SOAP Client that communicates with these services was installed on the Web server, outside the SOAP server. The WSDL specifications were generated by the Java2WSDL emitter. Tomcat and Axis are Java-based and freely available from the Apache Software Foundation. The services and the client were written in Java. All software is running on the Java 1.4 platform.

It is important to notice that when developing WS with Axis, the programmer doesn't need to bother about making WSDL interfaces or the actual encoding of the SOAP messages. All these will be automatically done by the system. Making all WS details transparent to the programmer enables him to focus only on the application logic implementation in Java and makes thus the system less error-prone.

The data transformations are implemented in Java using Jena [19]. These transformations replace the previous XSLT [20] data transformations implemented using Saxon [21], as Java transformations can exploit more of the RDFS semantics given by Hera models than the ones based on XSLT. For querying and constructing RDF(S) models we used ARQ, an implementation of SPARQL [11]. ARQ is now delivered as part of the Jena distribution kit.

### 3. Conclusion

In this paper we have described a distributed implementation for the Hera methodology. The implementation is based on WS due to the popularity and easy-to-implement features of WS. Because of the loose coupling of WS one can easily extend/replace a WS without affecting the other application services. Also these WS can be made available for other applications, for example the User Profile Service can be used by a Web querying application that wants to improve the accuracy of the returned results by making use of the user context. The Axis distribution kit proved to be a very flexible set of tools to support WS development. Users familiar with the Java programming language can seamlessly deploy Java methods as WS.

As future work we would like to extend the WSOA with new services like a data integration service responsible for integrating data from different, possibly heterogeneous data sources, or a data update service that allows for changes in the original data sources. In addition we would like to add a broker (based on UDDI) that will further decouple the communication between WS. Services will register themselves at the broker which will be responsible for forwarding requests and responses between WS. In this way the application doesn't need to know about WS location but only of the functionality these WS offer.

## References

- [1] Tomas Isakowitz, Michael Bieber, and Fabio Vitali. Web information systems. *Communications of the ACM*, 41(1):78–80, July 1998.
- [2] Daniel Schwabe and Gustavo Rossi. An object oriented approach to web-based application design. *Theory and Practice of Object Systems*, 4(4):207–225, 1998.
- [3] Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, and Maristella Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2003.
- [4] Nora Koch, Andreas Kraus, and Rolf Hennicker. The authoring process of the uml-based web engineering approach. In *First International Workshop on Web-Oriented Software Technology*, 2001. Available at: <http://www.dsic.upv.es/~west/iwwest01/files/contributions/NoraKoch/Uwe.pdf>.
- [5] Klaus-Dieter Schewe and Bernhard Thalheim. Reasoning about web information systems using story algebras. In *Advances in Databases and Information Systems (ADBIS 2004)*, volume 3255 of *Lecture Notes in Computer Science*, pages 54–66. Springer, 2004.
- [6] Flavius Frasinicar, Geert-Jan Houben, and Richard Vdovjak. Specification framework for engineering adaptive web applications. In *The Eleventh International World Wide Web Conference, Web Engineering Track*, 2002. Available at: <http://www2002.org/CDROM/alternate/682/>.
- [7] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tueke. The physiology of the grid: An open grid services architecture for distributed systems integration. Global Grid Forum, 2002.
- [8] Stefano Nativi. Service-oriented technology to support geosciences. In *Expanding Horizons: Using Environmental Data for Education, Research, and Decision Making Workshop*, 2003. Available at: [http://my.unidata.ucar.edu/content/Presentations/2003\\_expanding\\_horizons/nativioverview.pdf](http://my.unidata.ucar.edu/content/Presentations/2003_expanding_horizons/nativioverview.pdf).
- [9] Richard Vdovjak, Flavius Frasinicar, Geert-Jan Houben, and Peter Barna. Engineering semantic web information systems in hera. *Journal of Web Engineering*, 2(1-2):3–26, 2003.
- [10] Wireless Application Protocol Forum, Ltd. Wireless application group: User agent profile. 20 October 2001, 2001.
- [11] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. W3C Working Draft 20 February 2006, 2006.
- [12] Graham Klyne, Franklin Reynolds, Chris Woodrow, Ohto Hidetaka, Johan Hjelm, Mark H. Butler, and Luu Tran. Composite capability/preference profiles (cc/pp): Structure and vocabularies 1.0. W3C Recommendation 15 January 2004, 2004.
- [13] Zoltan Fiala, Flavius Frasinicar, Michael Hinz, Geert-Jan Houben, Peter Barna, and Klaus Meissner. Engineering the presentation layer of adaptable web information systems. In *Web Engineering - 4th International Conference (ICWE 2004)*, volume 3140 of *Lecture Notes in Computer Science*, pages 459–472. Springer, 2004.
- [14] Annrai O'Toole. Web service-oriented architecture: The best solution to business integration. Cape Clear Software, 2005. Available at: <http://www.capeclear.com/technology/clearthinking/websoa.shtml>.
- [15] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (wsdl) 1.1. W3C Note 15 March 2001.
- [16] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.1. W3C Note 08 May 2000.
- [17] Apache Software Foundation. Apache tomcat, 2006. Available at: <http://jakarta.apache.org/tomcat/>.
- [18] Apache Software Foundation. Webservices - axis, 2006. Available at: <http://ws.apache.org/axis/java/user-guide.html>.
- [19] Hewlett-Packard Development Company, LP. Jena - a semantic web framework for java, 2006. Available at: <http://jena.sourceforge.net/>.
- [20] Michael Kay. Xsl transformations (xslt) version 2.0. W3C Candidate Recommendation 8 June 2006, 2006.
- [21] Michael Kay. Saxon (the xslt and xquery processor), 2006. Available at: <http://saxon.sourceforge.net>.

# Adaptive Support of Inter-Domain Collaborative Protocols using Web Services and Software Agents

Adomas SVIRSKAS<sup>a,1,2</sup>, Michael WILSON<sup>b</sup>, Bob ROBERTS<sup>c</sup> and  
Ioannis IGNATIADIS<sup>c</sup>

<sup>a</sup> *Enterprise Communications Department, Institut Eurécom, France*

<sup>b</sup> *CCLRC Rutherford Appleton Laboratory, UK*

<sup>c</sup> *Kingston University London, UK*

**Abstract.** This paper explores the challenges of constructing an architecture for inter-organisational collaborative interactions based on Service Oriented Architecture (SOA), Web services choreographies and software agents. We present an approach to harmonisation of the "global" or neutral definition of business collaborations, with partner-specific implementations, which can differ in terms of platform, environment, implementation technology, etc. By introducing the concept of pluggable business service handlers into our architecture we draw on the work carried out by ebXML initiative, business services interfaces, in particular. Due to increasing need for better management of collaborative interactions, Virtual Organisations (VO) become an important tool for creation and maintenance of federated trust domains among the collaboration partners. We look into the software agents abilities to serve as the background support mechanism for the automation and management of the Virtual Organisations lifecycle.

**Keywords.** B2B collaborations, Web services, choreography, WS-CDL, agents

## Introduction

There is a growing need for flexible and adaptive business protocol support in Service Oriented Architecture (SOA) based e-business solutions and those based on Web services technology in particular. Recent developments in Web services field provide promising opportunities for integrating data, applications and business processes. The latter, however, is the most complex case of integration as it requires strong support for both business process semantics and technical infrastructure in order to tackle heterogeneity at all levels. According to Wombacher et al. [1], in today's B2B solutions landscape loosely coupled business processes are quite rare. Despite many promising advancements, Web services technology faces a number of issue to address heterogeneity at different levels of integration. Usage of simple stateless Web services is not sufficient for implementing business processes while static binding does not use full potential of loosely coupled systems and the SOA advantages [1]. In order for

---

<sup>1</sup>Corresponding Author: Adomas Svirkas, Enterprise Communications Department, Institut Eurécom, 2229 route des Crêtes - BP 193, 06904, Sophia Antipolis Cedex, France

<sup>2</sup>The work presented in this paper was done while working at CCLRC Rutherford Appleton Laboratory and Kingston University London, UK

distributed services to achieve common meaningful business goals, some rules of engagement are necessary. These rules govern the interactions between the services (referred to as services conversation) by defining message exchange sequences, message formats, roles of the collaboration participants etc. A set of service conversation rules is also referred to as business collaboration protocol or simply business protocol.

Business protocols can be made modelled using business process modelling tools and made available in machine-processable format through choreographies – declarative descriptions of service conversations. Vinoski [2] argues that Web services choreographies must take business processes into account: trivial Web services solve only trivial issues; non-trivial Web services must play a part in business processes. Business-to-business integration (B2Bi) requires standardized choreographies, i.e. definitions of the "conversations" between cooperating applications that allow them to work together correctly [2]. Simply put, choreography is a model of the sequence of operations, states, and conditions that control the interactions involved in the participating services [3]. The interaction prescribed by a choreography results in the completion of some useful function. Examples include the placement of an order, information about its delivery and eventual payment, or putting the system into a well-defined error state. Gortmaker et al. have presented an extensive of choreography definitions in [4] along with a thorough discussion of choreography and orchestration.

The rest of this paper is structured as follows. Section 1 discusses business protocol support issues and adaptivity dimensions, Section 2 explains the roles of choreography and orchestration in collaborative multi-party interactions, Section 3 provides a motivation for a business service handler-based approach to connect the public choreography to services and, in turn, private operations. Section 4 describes the main concepts of our approach and relates it to ebXML framework and Section 5 discusses applicability of software agent technology in dynamic Virtual Organisations (VO), followed by the conclusions section.

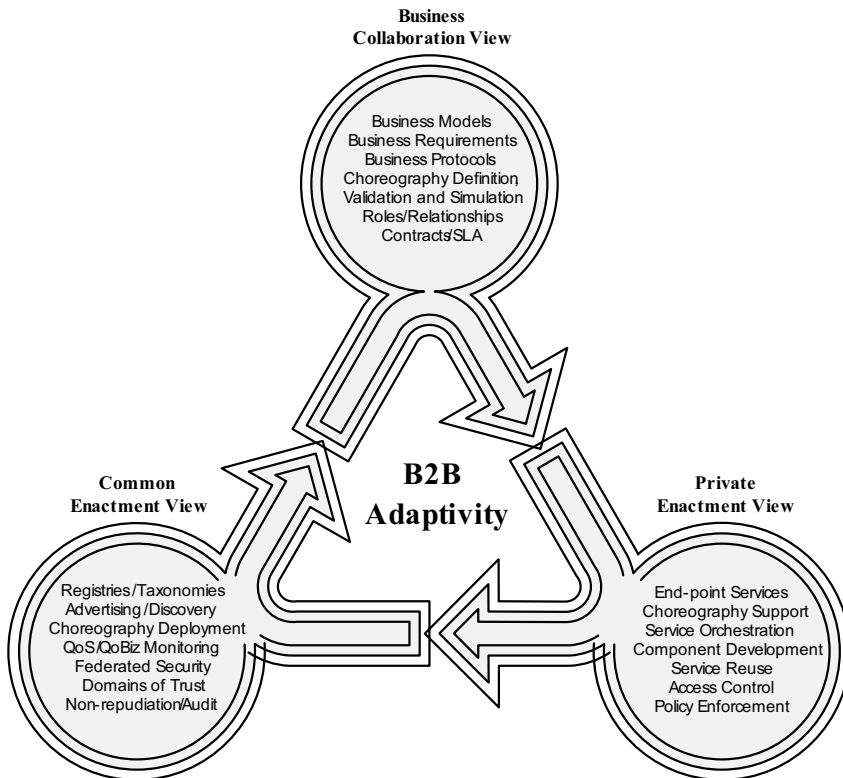
## 1. Business Protocol Support Issues

There are a number of requirements to be taken into account and satisfied in order to make the idea of commonly agreed business protocol specification and their subsequent enactment viable, let alone efficient, robust, scalable and secure. These requirements arise both from the business and technology domains and represent quite a wide spectrum of issues ranging from the field of activity of a business analyst to that of a software developer involved in business application coding.

Our paper aims to contribute to the aspect of adaptivity of business protocol support by collaborating partners. This aspect is a part of a broader issue of adaptivity in the business collaboration domain. Adaptation, which is defined as "*modification of an organism or its parts that makes it more fit for existence under the conditions of its environment*" [5], and the speed of it are crucial factors which determine the success and longevity of any business subject or formation in a constantly changing business environment. In the context of B2B collaborations adaptivity has several flavours: adaptivity of the business models to different business requirements and environments (Hofreiter & Huemer discuss this in [6]), adaptivity of business protocols in response to business models' changes, adaptivity of the partners' end-point services to the changes in the business protocols descriptions.

In addition, the business protocols should be adaptive to the changes of the partners enacting the roles defined in the protocols (both choreography and orchestration should support this). These requirements are contradictory in many cases and cannot be efficiently addressed in isolation: for example, optimisation of the work of a business analyst does not necessarily results in scalable implementations of end-point services and/or their faster time to market. Thus, a holistic approach is needed to business and collaboration modelling, enactment and process management in order to increase overall B2B collaboration adaptivity. Figure 1 depicts the circular dependencies between the main adaptivity dimensions and emphasizes the need for balance between the different views.

This kind of approach puts quite high requirements on the supporting IT infrastructure: not only the business modelling, choreography support, service development and deployment tools and established practices are needed, but also a coherent unifying framework, preferably based on open standards, should be established. Once the choreography definition is created, it needs to be deployed somewhere (the SOA suggests use of a registry) and advertised for reuse. It needs to be distributed to (or perhaps, depending on the overall interaction model, discovered by) the appropriate collaboration participants (dynamically chosen to play certain applicable roles) and accepted by each of them as a contract for subsequent interactions. The choreography script is then enacted by the participants' end-point services and this process should be monitored and managed by user and administrative tools. In addition



**Figure 1.** Relationships between different adaptivity views

to the basic requirements mentioned above, there are requirements for security and trust [7], [8], business-level contract [9] support, adequate levels of Quality of Service (QoS) and Quality of Business (QoBiz) by specifying, negotiating, monitoring and enforcement of Service Level Agreements (SLA) [10]. These and some other requirements are defining characteristics for B2B solutions and Virtual Organisations to be accepted by the corporate world [11].

It is therefore clear, that adaptivity in the business protocol support area is a complex issue and depends on many, frequently conflicting, factors. Cherinka et al. characterises complex adaptive systems in [12] as *“dynamically assembled systems characterized by multiple competing stakeholders, fluid requirements, emergent behaviour, and susceptibility to external pressures that can cause change across the entire system”*. This statement was used to reflect the nature of net-centric operations of the US Department of Defence; however it is applicable to the area of dynamic B2B Virtual Organisations, which also must accommodate unpredictable external factors that demand rapid response and flexibility to change [12].

## 2. Service Choreography vs. Orchestration

For the sake of clarity we would like to note the difference between choreography and orchestration, as these two terms are being used improperly sometimes, which causes confusion. Orchestration specifies the behaviour of a participant in a choreography by defining a set of "active" rules that are executed to infer what to do next, once the rule is computed, the orchestration runtime executes the corresponding activity(ies). Orchestration assumes existence of an entity, which is the central point of control and governs overall workflow of activities, effectively composing a new service from existing services. The standardization of orchestration and the emergence of a new application model will also benefit from a robust B2B layer, such as ebXML, in the Web services stack.

As a matter of fact, orchestration could take its full dimension from the extension of the business semantics to the application model [13]. Choreography, as explained before, is meant to be enacted by peers without an intermediary, at runtime, the choreography definition can be used to verify that everything is proceeding according to plan. Choreography can also be used to generate a public interface (e.g. abstract BPEL) that can be used to tie in internal activities to support the choreography [13]. Dubray also differentiates in [13] between the two concepts by arguing that choreography defines the fabric of an SOA while orchestration, helps to build "processing entities" - non-trivial services, which can perform tasks, needed to support complex business interactions.

Ross-Talbot [35] proposes a concept of behavioural backbone on top of a choreographed distributed infrastructure where the interactions among parties are described using a declarative language. It is possible to generate monitors for each of the roles, which allow gathering run-time information without affecting the behaviour of the services. The behavioural backbone provides the monitoring agents and the consolidation of that monitoring information against choreography. It allows observing non-intrusively potential collaboration deviations from the initial collaboration plan, which allows understanding the collaborative processes and managing them. Of course, such agents can serve many different purposes – security, transaction support, logging/auditing etc. Our approach based on business service handlers and application



level gateways is similar and takes into account the whole range of issues beyond monitoring. The software agent technology discussed in the Section 5 is in particular suitable for supporting such meta-protocols for monitoring and managing the collaborations.

To summarise this discussion, Figure 2 depicts different domains of control and inter-relation between the choreography and orchestration in business collaboration context.

We can distinguish four conceptually different domains here:

- *Administration domain* contains the resources belonging to an organisation or a large department, these resources are centrally managed according to corporate policies
- Within each administration domain typically exist several orchestration domains, which correspond to the established internal workflows and/or are used for service composition
- The goal of collaborating parties is to establish a federated collaboration domain (Virtual Organisation) using commonly understood interaction protocol policy-driven access control, trust management etc.
- One of the main instruments establish and maintain such federated collaboration domain (we can call this process VO management) is choreography support – from modelling of the interactions to deployment and enactment of the business protocols expressed by means of choreography descriptions. This is achieved by forming a choreography domain consisting of public services exposed by the collaboration partners. These services act as gateways to virtualised business services of the partners required to achieve the outlined business goals. In addition to passing the incoming messages during choreography enactment to the appropriate internal service endpoints, gateways are able to manage and monitor the interactions among the services by reasoning upon and enforcing the necessary policies. This style of interaction adheres to the managed public process e-business integration pattern [36].

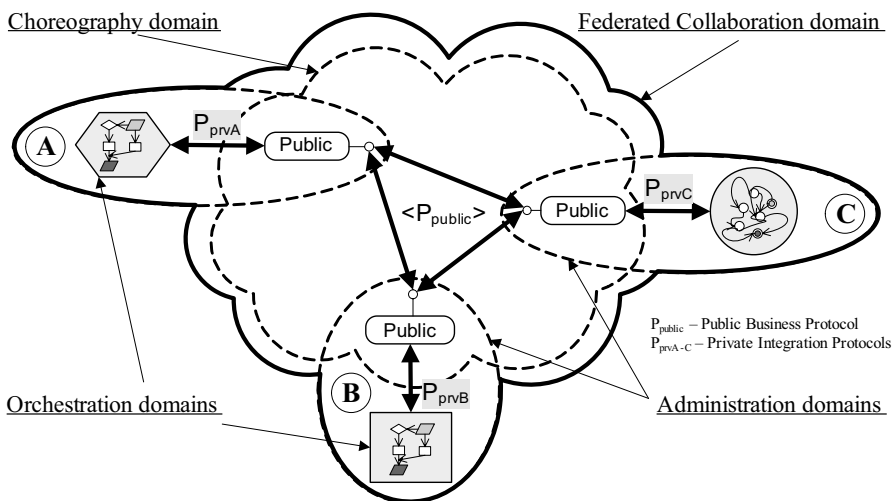


Figure 2. Web service choreography vs. orchestration

The considerations mentioned above and some earlier developments, such as Web Service Choreography Interface (WSCI) [14], ebXML Business Process Specification Schema [15] served as input for W3C to establish the Web Services Choreography Working Group and begin work towards a language that can be used to describe collaboration protocols of cooperating participants, which act as peers and their interactions may be long-lived and stateful. Web Services Choreography Requirements document [16], provides the following definition of choreography: "Web Services Choreography concerns the observable interactions of services with their users. Any user of a Web Service, automated or otherwise, is a client of that service. These users may, in turn, be other Web Services, applications or human beings. A specific set of interactions maybe related over time to some form of collaboration grouping that is initiated at some source and runs through a set of Web Services and their client. A choreography description is a multi-party contract that describes from global view point the external observable behaviour across multiple clients (which are generally Web Services but not exclusively so) in which external observable behaviour is defined as the presence or absence of messages that are exchanged between a Web Service and it's clients" [16]. A notable contribution to these requirements was [17], which advocated the need for complementary but separate languages - choreography programming languages and choreography description languages. Goland showed in [17] rather clearly, using motivating use cases, the difference (from choreography point of view) between executable languages such as Java, C#, BPEL and similar, and declarative description languages, which capture a global view of messaging activity and are not designed to provide information about how participants implement their individual tasks. Goland also explained in [17] the need for generating role-specific code skeletons from choreography description in order to facilitate faster and more convenient implementation of individual functionality. The choreography description language uses roles to differentiate between the participants in choreographies. We will discuss this aspect in greater level of detail in subsequent sections.

### **3. Adaptivity Issues in Choreography Life-Cycle**

The result of the mentioned W3C WS Choreography Working Group effort is WS-CDL language [18], which is the means to define a technical multi-party contract, mentioned above. WS-CDL specification is aimed at being able to precisely describe collaborations between any types of participants regardless of the supporting platform or programming model used by the implementation of the hosting environment, thus addressing heterogeneity issues [18]. Choreographies must also completely hide component-level implementation details. Moreover, the same choreography definition (potentially involving any number of parties or processes) needs to be usable by different parties operating in different contexts (industry, locale, etc.) with different software (e.g. application software) [18].

Choreography definition using WS-CDL allows building of more robust services because they can be validated statically and at runtime against a choreography description, verification absence of deadlocks and live-locks, etc. It also helps to ensure effective interoperability of services, which is guaranteed because services will have to conform to a common behavioural multi-party contract, mentioned earlier [19].

However, compliance of the participating services to the common contract might result that the choreography enactment is hard coded into the implementations of the

services and/or their composition mechanisms. This approach poses a two-fold problem: it reduces reusability of the services and also makes it difficult to change choreography description without a need for massive programmatic changes at the participating end-points. Goland [17] discusses these issues from developers' point of view in his contribution to WS Choreography requirements.

A possible alternative to global-contract choreography is a technique called mediation, where an intermediary agent is involved in communication between parties and ensures compliance of message flow to expected/requested behaviour of each party. A notable example of such approach is Web Service Modelling Ontology (WSMO) Choreography [20]. WSMO is an ontology for describing various aspects related to Semantic Web services [21].

The WSMO framework provides support for choreography and orchestration as part of interface definition of a WSMO service description. An interface describes how the functionality of the service can be achieved (i.e. how the capability of a service can be fulfilled) by providing a twofold view on the operational competence of the service: Choreography decomposes a capability in terms of interaction with the service (service user's view) Orchestration decomposes a capability in terms of functionality required from other services (other service providers' view) With this distinction different decompositions of process/capabilities are provided to the top (service requester) and to the bottom (other service providers). This distinction reflects the difference between communication and cooperation. The choreography defines how to communicate with the web service in order to consume its functionality. The orchestration defines how the overall functionality is achieved by the cooperation of more elementary service providers [22]. WSMO choreographies are based on the Abstract State Machines methodology. Cimpian & Mocan in their work [23] describe a process mediation approach based on WSMO choreographies and Web Service Execution Environment (WSMX) [23], a reference implementation for WSMO.

#### **4. Suggested Approach**

In this paper we propose software components called handlers to represent the logic of harmonizing public choreographed processes with private functionality of end-point services. Handlers are registered with and coordinated by a choreography support service, which, in turn, used by the end-point services to support global choreography contracts. Configured with pluggable handlers choreography support service mediates two-way message exchange between the "outside world" and the local processing entities. Based on the available handlers, various request types and formats can be routed, translated, and fulfilled by the business services. Choreography support service can relatively easily be reconfigured, adapted, and extended as new processing entities need to be supported. In addition, dynamic selection of processing entities to play the prescribed roles, policy enforcement, trust and security support and other non-functional tasks can be performed by the handlers. The handlers can be implemented as a chain of message filter put in front of processing entities deployments. A handler takes a choreography definition, the role, which processing entity is supposed to play and maps the choreography messages to local operations at run-time.

#### 4.1. The Role and Status of the ebXML

We call the handlers business service handlers, drawing a parallel with the naming used in ebXML framework [24]. The original name for these components in ebXML framework was Business Service Interface (BSI), which can be described as a piece of software that handles incoming and outgoing messages at either end of the transport [25]. The ebXML concept of a business transaction and the semantics behind it are central to predictable, enforceable e-business. It is expected that any Business Service Interface (BSI) will be capable of managing a transaction according to these semantics. Dubray explains the purpose of ebXML BSI in his overview of ebXML [24]: *“The Business Service Interface (BSI) should enforce the business collaboration protocol (ebXML BPSS). At any point in time, the BSI is able to determine if a message makes sense from a business perspective (is format correct? did it come on time? in the right sequence? ...). The BSI may be directly communicating with an application, but it is certainly wise to use a broker that will dispatch ebXML requests and responses to and from your business applications. Typically, this broker is going to be a business process management system.”* This explanation actually outlines the core functionality of BSI and justifies the need of pluggable brokers to support a variety of business process management systems. The OASIS ebXML Business Process (ebBP) Technical Committee (TC) later discussed a possibility to differentiate between Business Services Interface and Business Service Handler (BSH) in order to separate the abstract interface from its implementations. By proposing the name change to business service handler, the ebBP committee harmonised the naming between the business and messaging domains – the ebXML Message Service Specification [26] defines the Message Service Interface and Handler separately.

In ebBP Specification v 2.0 [27] the BSI is defined from a different perspective: as a logical definition for a party's actions, exposed as business services. It may be seen as a logical shared definition at different nodes. Logically, a BSI is a partner's implementation of the shared definition of business states and actions relevant to a common business goal. The BSI specifies the allowed set of business process and business object states of a business process, and the rules governing transitions between those states. In the context of the ebBP technical specification, only the shared business process is being managed. The interface to the BSI is through business messages and signals [27]. This defines the functionality of the BSI closely to the functionality of an individual partner required to support WS-CDL based common multi-party contract. Therefore the ebXML BPSS and W3C WS-CDL define substantially similar approach to enactment of common business goal and idea of pluggable business handlers follows this paradigm.

The ebBP technical specification does not, however, specify how the BSI is implemented. For example, the BSI may be enabled through a BSI-aware business application or through behaviour implemented as a part of a Message Service Interface component. The business application may business signals that are sent (realized) by the Message Service Handler [27]. Similarly, WS-CDL [18] does not specify how collaborating parties implement/map their services to comply with the common contract. We think that it useful to turn to the ebBP TC work when architecting choreographed Web Services solutions, as the ebBP v2.0 specification takes Web Services into account and explicitly relies on choreographed collaborations (no relation to WS-CDL is defined yet; the ebBP TC, however, is working on ebBP and WS-CDL layering). An ebBP Choreography is an ordering of Business Activities within a

Business Collaboration and is specified in terms of Business States and transitions between those Business States. Execution of the backend systems, which instruct the BSI to send or receive messages, advances the state of a collaboration. Similarly to WS-CDL, there is no execution engine associated to the collaboration itself. Although WS-CDL and ebBP address similar problem domains, the divergent foci of the two enables them to be layerable - while WS-CDL focuses primarily on the web service perspective, ebBP describes the pure business message flow and state alignment. As such they are not mutually exclusive.

Barreto [28] argues that WS-CDL and ebBP could be used in a loosely coupled, yet complementary manner, where WS-CDL supports the choreography based on endpoint references related to WSDL, while ebBP specifies the operation mapping to the recognized business transaction patterns. This association is beneficial and useful where complex activities occur in the collaboration environment.

#### *4.2. Operation Mapping*

One more notable aspect of ebBP v2.0 specification is mapping of Business Transaction patterns to abstract operations through the OperationMapping constructs (still work in progress) [27]. An operation mapping specifies a possible mapping of a business transaction activity to a set of Web Services operation invocations to enable the participation of non-ebXML capable business partner in an ebXML relationship. An ebBP definition does not itself contain a reference to a WSDL file, but rather references to operation names which can be deferred with specific WSDL files specified at the Collaboration Protocol Profile [25].

The goal of the operation mapping is to offer a flexible mapping scheme to map all document and signal interchanges to any combination of Web Services operation interactions. The mapping is also designed to define an operation mapping on both sides of a Business Transaction Activity (BTA). BTA represents the performance of a Business Transaction within a collaboration and is similar to WS-CDL interaction. This means that the ebBP specification can be used to define the abstract behaviour of complex collaborations between Web Services even in the case where no role in the collaboration is capable of ebXML [27].

The concepts found in ebBP v2.0 specification, therefore, provide several benefits for mapping choreography contracts to the end-point implementations:

- There are clear signs and concrete steps to support Web Services at WSDL operations level. The use of run-time correlation and endpoint references based on emerging addressing mechanisms such as WS-Addressing, WS-MessageDelivery etc. is recommended.
- The ebBP operation mapping is designed to support not only Web Services but other implementation techniques as well.
- The ebBP v2.0 specification suggests flexible approach to operation mapping (and common contract enforcement, in turn) by allowing operations to be mapped both in collaboration description and the partner's endpoints.

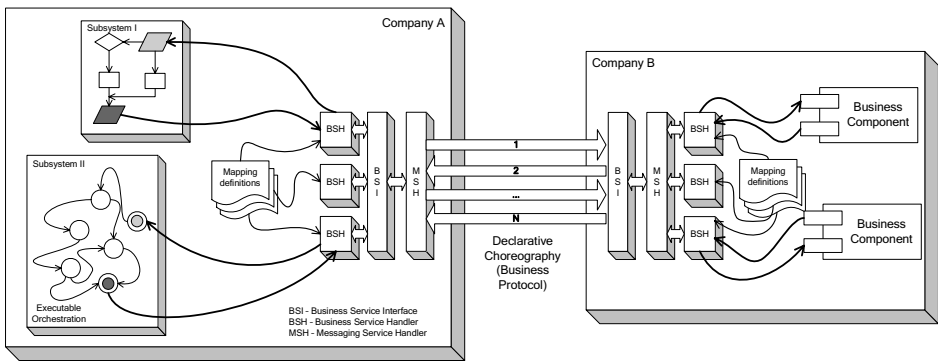
Therefore, ebBP [27] serves as a blueprint for architecting choreography-based solutions in general, not only ebXML compliant systems. The ebBP specification is being created (or closely watched) partially by the same individuals as WS-CDL specification, which creates synergy between the two efforts. We took into account many ideas described in ebBP specification and this will greatly help in our future work on this topic.

As we can see from the discussion above, changes in business protocol and choreography description may be quite costly for the end-points to support, therefore some intelligence is needed to allow smooth and fast propagation of the protocol changes. In this paper we propose to introduce an application-level gateway between the public business protocol (choreography) and the end-point composite services. This gateway is a part of the implementations of the collaboration participants' end-points. The functions of the gateway range from virtualisation of the business services of the end-point to choreography support via role-based decomposition of it and mapping of the incoming messages to the appropriate processing entities within the end-point.

As shown in Figure 3, Inside the gateway there is a set of software components called handlers, which perform distinct tasks, for example to represent the logic of harmonising public choreographed processes with private functionality of end-point services. There are two kinds of handlers – application-specific and application-independent [29]. The latter are reusable across the range of different applications, while the former depend on a particular system. Furthermore, the application-specific handlers are divided into supplier handlers and consumer handlers. Supplier handlers are triggered by the Reactor upon receiving an incoming message and their task is to find an appropriate consumer handler for the message to be processed. Locating of an appropriate consumer handler may be message content and choreography status based.

In order to organise the structure and the operation of the gateway component in a scalable and efficient manner, it is advisable to follow well established patterns for such applications. Schmidt has proposed usage of several patterns (Reactor, Component Configurator, Acceptor-Connector and other) in application-level gateways in [29]. Other patterns, such as chain of responsibility, also apply here.

The next section describes how choreographies and their adaptive support can be used in VO Management context.



**Figure 3.** Architecture of collaboration protocols support using gateways and handlers

## 5. Using Software Agents in Virtual Organisations Support

### 5.1. General Observations

Virtual Organisations (VO) are closely related to business collaborations between the services, organisations and individuals involved and are intended to facilitate, directly or indirectly, business solutions in most cases. VO management process can be perceived just as a type of business collaboration or (process) that uses the same mechanisms as for "operational" business collaborations (or processes). The collaboration agreement of a VO specifies processes related to the administration of the VO itself, such as changes to the VO membership or the collaboration agreement [11].

The software agent technologies are suitable mainly for domains of highly complex problems and systems with widely distributed information sources, domains with dynamically changing environment and problem specification, and for the integration of a high number of heterogeneous software systems [30]. Therefore, the agent technologies are suitable for usage in a Virtual Organisation, where the participants are geographically distributed, usually with heterogeneous software systems, and where the environment is dynamically changing in response to market needs and requirements.

(Software) agents are autonomous, which is very desirable in unknown scenarios (which usually tend to appear in the real world), where it is difficult to control directly the behaviour of complex business collaborations. Even though it is possible to encapsulate some behaviour by specifying "private" methods, agents must decide by themselves whether to execute their methods according to their goals (agents must be proactive), preferences and beliefs. Agents are also flexible, they have to learn from, and adapt to, their environment. This is important, since when designing an agent system, it is impossible to foresee all the potential situations that a particular agent might encounter, and specify agent behaviour optimally in advance. This kind of situation is highly probable in the most of non-trivial VO interactions.

In a VO setting, a multi-agent system can be employed for supporting internal processes (intra-enterprise level) of the enterprise (e.g. planning and control, resource allocation, production process simulation, and on the other hand, it can support cooperation and negotiation among enterprises (extra-enterprise level) across a value chain (e.g. customers, suppliers, material and service providers, etc). Both types of agents can coexist in an organisation.

In addition to the external-internal dimension of agents' classification, there is another one, which is based on the *specific purpose* of the agent services. The reason for making this distinction explicit is the fact that the business services themselves can be considered as agents as they satisfy most of the agents' characteristics. Maximilien & Singh [31] in their work of cataloguing Web services interaction styles argue that viewing services as agents enables us to augment the interaction styles of Web services as interactions between and among service provider agents and service consumer agents. Therefore, it is important to denote the areas of responsibility of the business services and the supporting agents.

As we have explained above, Web services are characterized not only by an interface but also by the business protocols (choreographies) they follow. While business protocols are application specific, much of the software required to support such protocols can be implemented as generic infrastructure components Alonso et al. For example, the infrastructure can a) maintain the state of the conversation between a

client and a service, b) associate messages to the appropriate conversation, or c) verify that a message exchange occurs in accordance to the rules defined by the protocols (for example WS-CDL). Part of the task of the infrastructure is also the execution of meta-protocols, which are protocols whose purpose is to facilitate and coordinate the execution of business protocols. It is convenient to think of the agents as the meta-protocol enablers, paving the way for the business services.

For example, before the actual interaction can begin, clients and services need to agree on what protocol should be executed, who is coordinating the protocol execution, and how protocol execution identifiers are embedded into messages to denote that a certain message exchange is occurring in the context of a protocol. In the Web Services domain, WS-Coordination is a specification that tries to standardize these meta-protocols and the way WSDL and SOAP should be used for conveying information relevant to the execution of a protocol [32]. In the Multi-Agent System (MAS) domain, there are other protocols for agents' interaction, which can be useful for implementing the meta-protocols.

Having distinguished between the agents and the services, we need to make sure that these two types of entities coexist peacefully within a single architecture and interoperate properly. Maximilien & Singh [33] propose a framework that augments a typical Service-Oriented Architecture (SOA) with agents. Their principal idea is to install software agents between service consumers and each service that they consume. These consumer service agents expose the same interface as the service. However, they augment the service interface with agent-specific methods. The consumer communicates its needs via the augmented agent interface. Service method invocations are done via the service agent who, in turn, monitors and forwards all calls to the selected service. Both business and meta-protocols can be modelled, validated and verified using the WS-CDL language and tools.

A good example of a consistent set of meta-protocols is the VO Management domain, where the business collaboration partners (peers) interact by the rules agreed by all the VO members and VO managers. These rules are enacted partly by direct interaction between the peers, and partly by the peers and the VO Management. We discuss Virtual Organisation Management in the next section.

The concept of multiple agents can also be useful in general-purpose Web service composition. Maamar et al. [34] present an agent-based and context-oriented approach that supports the composition of Web services. To reduce the complexity featuring the composition of Web services two concepts are put forward in their work, namely, software agent and context. During the composition process, software agents engage in conversations with their peers to agree on the Web services that participate in this process. Conversations between agents take into account the execution context of the Web services. The security of the computing resources on which the Web services are executed constitutes another core component of the agent-based and context-oriented approach presented by Maamar et al [34].

## *5.2. Virtual Organisation Management*

A VO typically follows the life cycle consisting of the four phases: a) Identification (Opportunity Identification and Selection), b) Formation (Partner Identification and Selection, and Partnership Formation), c) Operation (Design, Marketing, Financial Management, Manufacturing, Distribution), and d) Termination (Operation Termination and Asset Dispersal). The management of a VO through those phases can



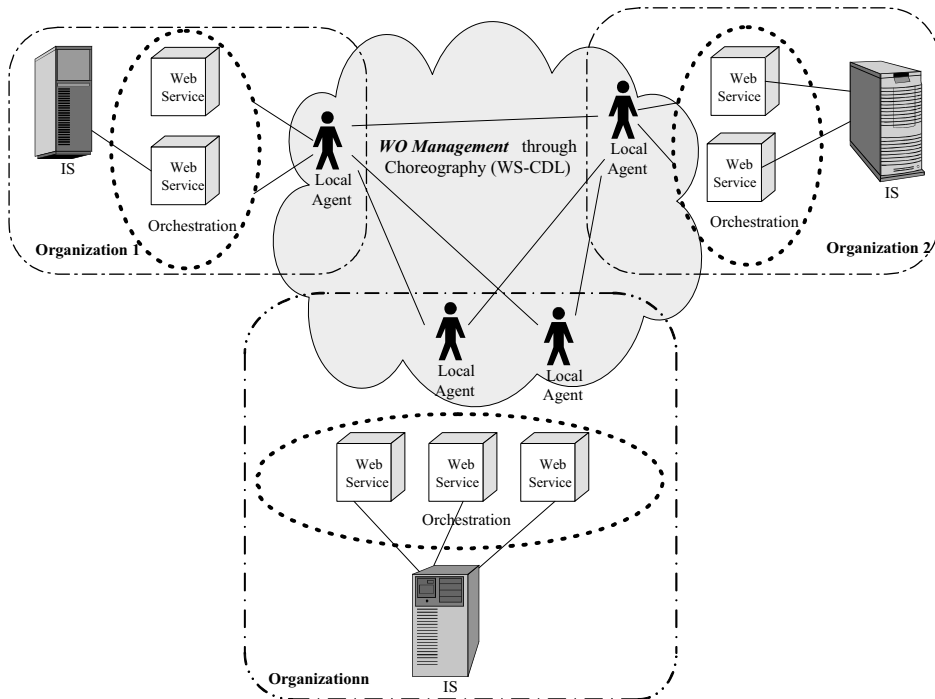
be described just as a type of business process (or collaboration) that uses the same mechanisms as for “operational” business processes (or collaborations). The collaboration agreement of a VO then specifies the processes that are related to the administration of the VO itself, for example changes to the VO membership [11].

With regard to the software agent technology, it can bring various advantages to the domain of management of e-collaborations. The technological and integration aspect is covered by the Foundation for Intelligent Physical Agents (FIPA) [38], which tries to maximize interoperability across agent-based applications, services and equipment.

Figure 4 illustrates our proposal for the management of Virtual Organisations using Web Service Choreography and Software Agents. The innovation lies in the fact that whereas Web Service Choreography can be used to coordinate the interactions between Web Services and their consumers, software agents can be inserted in front of those services, and their actions choreographed.

Within a Virtual Organisation, intelligent software multi-agents can take some of the load in each of the phases of identification, formation, operation and termination of a VO, by automating the relevant processes. Various agent technologies can also be used for the agents’ private knowledge, maintenance, specification of various ontologies, and ensuring service interoperability across the value chain.

The proposed solution is a generic one, in the sense that it does not distinguish between the number of agents or their type (e.g. per service, business process, or enterprise). It assumes, however, that at least one local agent exists per each



**Figure 4.** Virtual organisation management with choreography and agents

organisation that participates in a VO. The interaction between the organisations in the VO is carried out with interactions between the respective agents. The latter communicate with the Information System (IS) of the organisation via the appropriate Web Services. Whereas within a single organisation those Web Services follow orchestration rules, as described earlier, the whole VO is coordinated by choreography rules that are enacted by each of the local agents assigned to an organisation. In that process, agents communicate and exchange information with local agents of other organisations. The use of agents adds flexibility to the operations of the VO, whereas at the same time the use of choreography rules ensures the efficient management of a VO without the need for a centralised service.

The process of the creation of a Virtual Organisation has its counterpart in the cooperative team creation or coalition formation processes in the agent technologies domain. In this domain, a group of cooperating agents (coalition) is formed to fulfil a common goal. The individual agents are self-oriented and they don't share all information or their intentions. The agent technologies in this case classify the knowledge as public, private and semi-private. This has a high potential for the management of Virtual Organisations, where there is not a central point of control, but the e-collaborations are rather peer-to-peer, in which case it is important for each peer to have control over the availability of its own information to the other peers in the network and restricting access to the confidential data.

## **6. Conclusions**

In this paper we have proposed an approach to the issue of choreography support in heterogeneous collaborative business interactions. We base our concepts on the assumption that service choreographies can be mapped to end-point operations using either rich service universal descriptions or end-point specific operation mappings. In both cases it is possible to derive programmatically the configuration artefacts, which can be used to configure business service handlers dynamically at the runtime. This possibility is attractive from many points of view, the most important perhaps being clean separation of business services interfaces and business services implementation.

Business service handlers can also be used for various other purposes – policy enforcement, trust and security support, collaboration correctness monitoring, QoS monitoring, transaction logging, etc. Choreography languages, such as WS-CDL can perhaps be enhanced to support declarative specification of the mentioned aspects for subsequent programmatic propagation of these specifications to the service end-points and mapping to the end-point specific mechanisms.

These ideas make basis for future research in this area alongside with detailed design of the pluggable business handler framework, which is described in this paper at conceptual level and many decisions still need to be made. Standard operation mappings between choreographies and implementation languages such as Java, C#, BPEL are one of the main issues in this area along with rich service description and matchmaking problems. It is a promising sign that the ebXML BPPS specification takes into account these issues and drives the effort to solve them in standard interoperable manner, as support of the open standards is crucial for adoption of solutions.

In the context of our work we use Web Services as the underlying technology to implement of models. The requirements for VO management described here can be met

by a subset of the current WS\* specifications, but they require secure, stable and interoperating implementations from a variety of IT vendors, which is not the case yet.

We also believe that combination of SOA and Web services in particular with the software agents can be very promising. The agents differ from services in a number of ways, they are necessary when users specify the goals they wish to achieve, rather than the actions they need to expose or perform (where the services fit well). Therefore usage of the agents may prove to be beneficial in non-trivial areas requiring rich interaction such as service matchmaking, contracting support, trust management etc.

## Acknowledgements

The results presented here are partially funded by the European Commission under contracts IST-2003-01945 and IST-2005-027169 through the projects TrustCoM [39] and PANDA [40]. The authors would like to thank members of the partner organisations working in these projects: SAP, ETH Zurich, European Microsoft Innovation Centre, Imperial College London, SICS, Kings College London, University of Milano, University of Kent, BAE Systems, BT, IBM, Q-PLAN N.G., Altec S.A., Czech Technical University of Prague and others.

## References

- [1] Wombacher, A. et al., Matchmaking for Business Processes Based on Choreographies. Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), pp. 359-368.
- [2] Vinoski, S., The Truth about Web Services. Web Services and Component Technologies. 2001. Available from: <http://www.proinfo.com.cn/chinese/menu2/3/102501.ppt>.
- [3] Web Services Architecture. W3C Working Group Note. 11 February 2004. Available from: <http://www.w3.org/TR/ws-arch/>.
- [4] Gortmaker, J., Janssen, M., and Wagenaar, R., SOBI Business Architectures and Process Orchestration: Technical Overview. Telematica Instituut, Enschede, The Netherlands, 2004.
- [5] Merriam-Webster Online Dictionary. Available from: <http://www.m-w.com>.
- [6] Hofreiter, B. and Huemer, C., Registering a Business Collaboration Model in Multiple Business Environments. Proceedings of the OTM Workshop on Modeling Inter-Organizational Systems (MIOS 2005) Larnaca, Cyprus, 2005.
- [7] Arenas, A., Djordjevic, I., Dimitrakos, et al., Toward Web Services Profiles for Trust and Security in Virtual Organisations. Proceedings of the 6th IFIP Working Conference on Virtual Enterprises (PRO-VE 2005), Valencia, Spain, 2005.
- [8] Jøsang, A. Keser C., and Dimitrakos, T., Can We Manage Trust? Proceedings of the Third International Conference on Trust Management (iTrust), Rocquencourt, France, 2005.
- [9] Dimitrakos, T., Djordjevic, I., Milosevic, Z., et al., Contract Performance Assessment for Secure and Dynamic Virtual Collaborations. Proceedings of the EDOC 2003, Brisbane, Australia, 2003.
- [10] Morsel, A., Metrics for the Internet Age: Quality of Experience and Quality of Business. Hewlett-Packard Labs Technical Report HPL-2001-179, 2001.
- [11] Svirskas, A., Wilson, M., Arenas, et al., Aspects of Trusted and Secure Business Oriented VO Management in Service Oriented Architectures. Workshop Proceedings of Seventh IEEE International Conference on E-Commerce Technology, IEEE Computer Society, Munich, Germany, 2005.
- [12] Cherinka, R., Miller, R., and Smith, C., Beyond Web Services: Towards On-Demand Complex Adaptive Environments. MITRE Technical Paper, 2005.
- [13] Dubray, J. J., WS-CDL - Choreography Description Language. Available from: [http://www.ebpm.org/ws\\_-\\_cdl.htm](http://www.ebpm.org/ws_-_cdl.htm).
- [14] Web Service Choreography Interface. W3C Note. 8 August 2002. Available from: <http://www.w3.org/TR/wsci/>.

- [15] ebXML Business Process Specification Schema. Version 1.01. Available from: <http://www.ebxml.org/specs/ebBPSS.pdf>.
- [16] Web Services Choreography Requirements. W3C Working Draft. 11 March 2004. Available from: <http://www.w3.org/TR/ws-chor-reqs/>.
- [17] Goland, Y. Y., A proposal for W3C Choreography Working Group Use Cases & Requirements. 16 May 2003. Available from: <http://lists.w3.org/Archives/Public/www-archive/2003May/att-0029/chor.htm>.
- [18] Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation 9 November 2005. Available from: <http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/>.
- [19] Ross-Talbot, S. and Brown, G., Dancing in Time with the Services. WS-CDL. 1st Feb 2005 NY Java SIG, NYC, USA. Available from: <http://www.javasig.com/Archive/lectures/JavaSIG-CDL-SRT.ppt>.
- [20] Roman, D. Scicluna, J., and Feier, C. (eds.), Ontology-based Choreography and Orchestration of WSMO Services. WSMO Final Draft 1 March 2005. Available from: <http://www.wsmo.org/TR/d14/v0.1/>.
- [21] Feier, C. (ed.), WSMO Primer. WSMO Final Draft 01 April 2005. Available from: <http://www.wsmo.org/TR/d3/d3.1/v0.1/>.
- [22] Roman, D and Lausen, H. (eds.), Web Service Modeling Ontology – Standard (WSMO Standard). WSMO Working Draft 8 July 2004. Available from: <http://www.wsmo.org/2004/d2/v03/>
- [23] Cimpian, E. and Mocan, A., WSMX Process Mediation Based on Choreographies. 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management (BPM 2005), September 2005, Nancy, France.
- [24] Dubray, J. J., ebXML. Available from: <http://www.ebxml.org/ebxml.htm>.
- [25] Enabling Electronic Business with ebXML. OASIS Whitepaper. 2000. Available from: [http://www.ebxml.org/white\\_papers/whitepaper.htm](http://www.ebxml.org/white_papers/whitepaper.htm).
- [26] ebXML Message Service Specification. Version 2.0 rev C. 21 February 2002. Available from: [http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS\\_v2\\_0rev\\_c.pdf](http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0rev_c.pdf).
- [27] ebXML Business Process Specification Schema v2.0. Working Draft 11, 5 April 2005 (Committee Draft candidate).
- [28] WS Choreography WG conference call 15 March 2005. Minutes. Available from: <http://www.w3.org/2002/ws/chor/5/03/15-minutes.html>.
- [29] Schmidt, D. C., Applying a Pattern Language to Develop Application-level Gateways. Design Patterns in Communications, L. Rising (ed.), Cambridge University Press, 2000.
- [30] Jennings N. R. and Bussmann S., Agent-Based Control Systems. IEEE Control Systems Magazine 2003, 23:3, 61-74.
- [31] Maximilien E. M. and Singh M. P., Multiagent System for Dynamic Web Services Selection. 2005
- [32] Alonso, G., Casati, F., Kuno, H., and Machiraju, V., Web Services Concepts, Architectures and Applications. Springer-Verlag, 2004.
- [33] Maximilien E. M. and Singh M. P., Toward Web Services Interaction Styles. Proceedings of the 2005 IEEE International Conference on Services Computing (SCC'05), Orlando, Florida, USA, 11-15 July, 2005, pp 147-154.
- [34] Maamar, Z., Mostefaoui, S. K., and Yahyaoui, H., Toward an Agent-Based and Context-Oriented Approach for Web Services Composition. IEEE Transactions on Knowledge and Data Engineering 2005, 17:5, 686 - 697.
- [35] Ross-Talbot, S., ACM Queue - a Conversation with Steve Ross-Talbot on the Role of Choreographies and Pi-Calculus, Available from: <http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=370&page=4>.
- [36] Adams, H. et al., Best practices for Web services: Part 4, A Managed Public and Private Process Application Pattern Scenario. 2002. Available from: <http://www-106.ibm.com/developerworks/library/ws-best4/?n=ws-12122>.
- [37] Strader, T. J., Lin, F., and Shaw, M. J., Information Structure for Electronic Virtual Organization Management. Decision Support Systems 1998, 23, 75-94.
- [38] Foundation for Intelligent Physical Agents (FIPA). Available from: <http://www.fipa.org>.
- [39] Svirskas, A., Arenas, A., Wilson, M. D., and Matthews, B., Secure and Trusted Virtual Organization Management. ERCIM News No. 63, October 2005, SPECIAL: Security and Trust Management, 2005.
- [40] Ignatiadis, I., Roberts, R., and Svirskas, A., Operational Support in Virtual Enterprises through Collaborative Process Automation. Proceedings of the 1st International BIOPoM 2006 Conference, London, 2006.

# Using a Rule Language for Capturing Semantics in Web-Based Systems

Tanel TAMMET<sup>a</sup>, Hele-Mai HAAV<sup>b,1</sup>, Vello KADARPIK<sup>a</sup> and  
Marko KÄÄRAMEES<sup>a</sup>

<sup>a</sup>*Institute of Computer Science, Tallinn University of Technology, Estonia*

<sup>b</sup>*Institute of Cybernetics at Tallinn University of Technology, Estonia*

**Abstract.** Development of web-based applications needs tools in order to make the application development more effective by using/re-using business rules as well as web services. These tools should incorporate facilities for capturing different semantic aspects of an application. This paper presents a new conceptual and technological framework of using a rule language and rule engine for capturing semantics in modern web-based systems. The framework enables to cover two aspects of semantics in web-based systems: business rules and web service composition logic. The technology consists of 2 main parts: the application server Xstone for creating 3-layered systems and the RqlGandalf rule solver. The middleware server Xstone connects to Oracle, PostgreSQL databases and the RqlGandalf rule system. The RqlGandalf rule system is targeted for two different tasks: defining and using business logic rules as well as for rule-based synthesis of complex queries over web services. The presented rule-based system development technology is implemented for the Linux platform as open source software.

**Keywords.** Rule-based systems, web-service composition logic, business rules

## Introduction

Development of web-based systems needs tools in order to make the application development more effective by using/re-using business rules as well web services. These tools should incorporate facilities for capturing different semantic aspects of an application. Currently, semantic technologies are gaining more and more importance, mainly with respect to systems of distributed artificial intelligence including semantic web applications.

One aspect of semantics of web applications can be considered as business rules. As current tools that employ rule engines follow basically paradigm of expert systems, then there is need for more general approach. Rule languages used in current systems are very simple, basically intended to write if-then rules as in RuleMachines VisualRule Studio [1], Mindbox ARTEnterprise 10.0 [2], and ILOG Jrules [3].

Another aspect of semantics of web-based systems can be considered as rules used for composition of web services. Web service developers spend a lot of time manually discovering and composing web services to meet the requests for new services. There is lack of practical tools for supporting developers in semantic web service modeling and composition.

---

<sup>1</sup> Corresponding Author: Institute of Cybernetics at TUT, Akadeemia 21, 12618 Tallinn, Estonia; E-mail: helemai@cs.ioc.ee

In this paper, we present a new conceptual and technological framework for using a rule language and rule engine for capturing semantics in modern web-based systems. The framework enables to cover two aspects of semantics in web-based systems: business rules and web service composition logic.

The technology consists of 2 main parts: the application server Xstone for creating 3-layered systems and the RqlGandalf rule solver. The middleware server Xstone connects to Oracle, PostgreSQL databases and the RqlGandalf rule system. The RqlGandalf rule system is targeted for two different tasks: defining and using business logic rules as well as for rule-based synthesis of complex queries over web services.

The RqlGandalf rule solver is based on the first-order logic automated deduction system Gandalf [4]. Despite the common core, the rule systems in the framework come in two layers:

- The rule system, which answers queries about data based on logical rules input by the user or a programmer. In some sense Rql rule system is analogous to SQL systems, but the Rql end-user language allows the user to write complex logical rules not allowed by SQL. While SQL is targeted for manipulating and querying data, Rql rule system is targeted for defining and using business logic rules.
- The RqlGandalf rule solver has built-in capabilities for automatically creating (synthesising) programs for solving queries, under the assumption that data is input from web services. In other words, it finds out suitable web services and combines them in a suitable way. The program constructed by the RqlGandalf solver calls the services, loops over the results, selects fields, etc.

The whole framework and technology are called Rql system in this paper. All the core parts of the system are written in ANSI C and use the standard GNU tools for configuration, compilation and installation. The XSTONE/RQL technology suite is covered by GPL. Pilot applications of the proposed framework are developed on the basis of Data Exchange Layer X-Road of Estonian State Information System [5] and on a portfolio management system of an assets management company. The Rql system is implemented for the Linux platform as open source software and is available at [6]. As this paper is concentrated to providing the overall framework, then most of implementation details are omitted. For implementation details we refer to our previous works in [7, 8], and Rql system web site [6].

The technology is intended to be useful for developing web applications for the following domains: banking, insurance, capital markets, telecom fault detection, diagnostics, e-government, etc.

The rest of the paper is structured as follows. Section 1 gives overview of the approach proposed and introduces Xstone server. Section 2 presents how to define and use business rules by RqlGandalf rule system. Web-service composition component is described in Section 3. Related works are discussed in Section 4. Section 5 presents some conclusions and future work.

## 1. General Approach Used for the Framework

### 1.1. Overview of Rql System Approach

The Rql technology provides a mid-layer between a database and interfaces to other systems: both the (human) user interface and interfaces to software systems. The

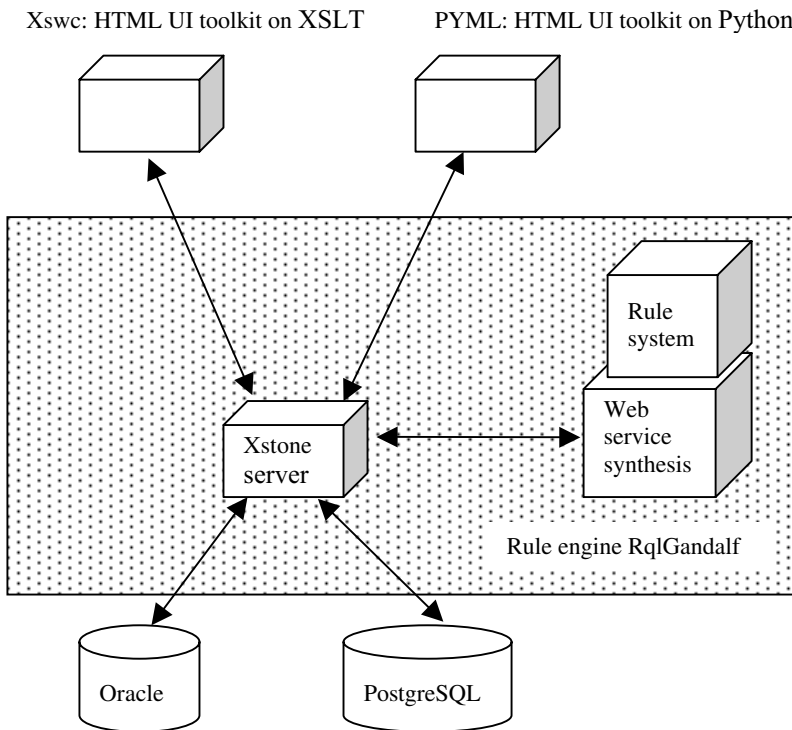
system is particularly useful for cases where the business logic must have numerous (possibly frequently changing) interfaces.

The basic idea of using the Rql technology is to write the business logic (database queries, rule queries, other operations) and web service composition semantics as a part of the mid-layer (i.e. as a part of Xstone server), keeping the interface code separate from this mid-layer.

The Rql technology consists of the following 3 parts:

- Xstone server that is the application server for creating 3-layered systems.
- RqlGandalf rule solver. It is able to read data from an SQL database: it can call Xstone and import/convert SQL-given data into its own knowledge base. Xstone is able to call the Rql system analogously to calling SQL servers, and outputting Rql-computed results in the XML format.
- Tools for building user interfaces. Xswc is the system for building browser-based user interfaces for Xstone, capable of obtaining and manipulating data by calling Xstone. In addition, there is the Python-based toolkit PYML (Python Markup Language) for building browser-based user interfaces against the functionality programmed into Xstone server.

The following Figure 1 shows all three layers of the Rql technology in their interaction.



**Figure 1.** Components of Rql technology

In the following, each part of the Rql technology is described in more detail.

### *1.2. The Xstone Server - a Kernel of Rql System*

Xstone is a middleware server useful for developers creating complex server-based applications. It is relatively simple and highly efficient. It is easy to extend and modify for developers needs. The server is written in C language.

Using the Xstone server means - for most cases - adding and changing SQL queries, XML definitions and logical business rules, without actually changing the C code. Xstone currently contains adapters for Oracle, PostgreSQL databases and the Rql rule system. Xstone also supports WSDL and SOAP standards for web services. Xstone answers CGI-protocol queries over HTTP, returning answers in XML, CSV or tuple formats. Xstone is called from application/interface programs using HTTP with the get protocol. The default output of Xstone (i.e. answers to HTTP queries) is in the XML format, but several different output formats are available as well.

Xstone functions perform similarly to CGI programs and they need to be declared in Xstone function file in XML format. The function declarations indicate the number and usage of HTTP parameters and the full text of the SQL or ELM language (input language of Rql rule system) queries/procedures to be performed.

The Xstone function file is a plain list of function definitions in XML. More information about function file can be found in [6]. For more complex needs, it is possible to add C functions to Xstone. The functions added are called via the definitions in the function file. In this case, recompilation of Xstone is needed.

### **Xstone Query Example**

Xstone query is ordinary CGI without argument names as follows:

```
http://www.xyz.com:8888/xstone/myquery?john&21
```

The answer to the query is an XML table created from the SQL query result as in the following case, where SQL query returns all customer with first name John and with age more than 21:

```
<data>
<rec>
<fname>John</fname><sname>Clark</sname><age>
>32</age>
</rec>
...
<rec>
<fname>John</fname><sname>Smith</sname><age>
>24</age>
</rec>
</data>
```

### *1.3. Tools for Building User Interfaces*

There are two sets of tools for building user interfaces in Rql technology as follows.



### 1.3.1. The xswc Toolkit

The Xstone server web client xswc is a toolkit for building browser-based user interfaces against the functionality programmed into Xstone. This package is primarily to be used as a prototype (template) for creating web (browser-based) clients for the Xstone server.

Web client xswc contains two main parts: C code for the xswclient CGI program and XSL files for the automatic, yet highly configurable XML-to-HTML conversion mechanism. The XSL files could be used completely separately from the xswclient CGI program.

### 1.3.2. PYML Toolkit

PYML is the Python-based toolkit for building browser-based user interfaces against the functionality programmed into Xstone. PYML is a system for integrating HTML and Python. In essence, it allows writing Python code directly into HTML, much like PHP, if the actual PHP code is written directly into HTML. PYML is designed for creating dynamic web pages, especially for database applications.

## 2. Capturing Business Rules

The Rql rule system provides 2 languages for representing rules. They include the Extended Logical Markup language (ELM) [9] for nontechnical users. With this language, business rules can easily be represented with business terms and automatically translated for execution by the rule solver RqlGandalf [10].

For system developers, there is rich and expressive language the Extended Common Logic (ECL) [11] used to specify rules for the rule solver RqlGandalf. In this section, the Rql rule system languages and the RqlGandalf rule solver are introduced.

### 2.1. The ELM Language

The input language of the Rql rule system for nontechnical users is the ELM language [9] that is a sugared version of extended first order predicate calculus (FOL). Extensions to FOL allow writing predefined computable functions, sentences in SQL and several nonmonotonic and (weak) epistemic operators, as well as select data sources to be used in queries.

The language syntax is influenced by N-triples and N3 syntaxes of RDF [12] as well as DATALOG and PROLOG. It allows to write predicates with arity one and two in an infix notation (e.g. *John is emitent*), while predicates of higher arity are written in prefix notation as e.g. *instrument(ticker, investmenttype, emitent, emitentcap)*.

ELM is a kind of a language called “controlled natural language”: a formal language aimed at resembling natural language, hence being human-readable without special training or IT background, and writeable with short training. The formal definitions for the language can be found in [9]. The ELM language is transformed to the extended FOL, which is internal language of the Rql rule solver. Most important extensions to ordinary FOL are special higher order query functions, for example \$sumforallfacts, \$maxforallfacts, etc. These are used to express sum, max, average, etc of data obtained from derivable facts. These functions are widely used in business applications. Above mentioned types of functions loop over all the given or derivable

facts, which correspond to criteria given in the function term and compute sum (or max or etc) of all the integers in a given place.

Specific business rules often use special extralogical query functions in addition to standard FOL as explained above. For example, in the portfolio management domain, the following business rule can be used:

```

if (($depositpercentage is <= to $maxdepositpercentage) and
    ($interestpercentage is <= to $maxinterestpercentage) and
    ($stockpercentage is <= to $maxstockpercentage))
then
    okassetsdistribution.

```

The rule above is used for checking assets distribution of a client according to maximum percentages for deposit, interest and stocks given by the user (e.g. portfolio manager). The user may relatively often change business rules as new queries, constraints, etc appear.

## 2.2. The RqlGandalf Rule Engine

The core of the Rql rule system is the RqlGandalf rule solver: it reads in text in the ELM language containing data, rules and queries, and then attempts to find answers to queries. When answers are found, it outputs an XML structure analogous to Xstone output for SQL queries (see previous section).

One major difference from SQL is that the rule system can be contradictory (RqlGandalf rule solver will detect that), and when answers are found to queries, then RqlGandalf will also create a step-by-step explanation for each answer: which facts and rules were used for finding the answer, what were the intermediate steps.

The Rql rule system is used in Xstone similarly to how Xstone uses the Oracle or PostgreSQL database system: Xstone users write SQL function bodies in Xstone XML function definitions, and in the similar way one can write ELM data definition, rule and query language for rule-based queries.

This works in two ways as follows:

- Each time an Rqlquery function is called via Xstone, Xstone will call RqlGandalf with both the query and the Xstone flag, causing RqlGandalf to read in additional data and rules from the database.
- RqlGandalf assumes that all the rules and data it should thus read in are present in the database table rqlrule, in the body field. It reads in all these fields from the Oracle or PostgreSQL database, assuming each field contains ELM text.

The RqlGandalf rule solver used in the Rql technology is an extension of Gandalf [4] that is a resolution theorem-prover for first-order classical logic with equality in order to make it applicable in rule-based and web-based IS development. It is well known that FOL theorem provers like Gandalf are optimized for speed and efficiency but should be extended to be useful for practical applications. Most theorem provers are aiming to solve small but hard problems as in mathematics. The RqlGandalf shows one direction to how FOL theorem provers can be adapted towards business and web-based applications. In practical tasks it is common to have a hard situation where a proof task contains both an “ordinary” FOL part well suited for resolution and a large special part not well suited for resolution. In [10] it is proposed a simple “black box”

system called “chain resolution” for extending standard resolution in order to significantly improve performance of solver in the context of problems involving large ontologies and terminological reasoning as in many business applications. It is shown in [10] that chain resolution is sound and complete.

The proposed rule-based technology has been first applied to creation of a portfolio management system prototype in an asset management company [6]. This is a web-based system for portfolio management, where one could add/modify data about stocks and other financial instruments, add/modify portfolios and their content, etc. The system contains a set of modifiable rules on the portfolios and allows performing fixed and free form queries on the portfolios, basically checking whether a portfolio satisfies rules input into the system.

### 3. Rule-Based Web-Service Composition

#### 3.1. Composition of Web Services

Our logic-based web-service composition approach is based on general scheme of automated deduction and program synthesis [13, 14]. According to that, we formulate an axiomatic theory of the application domain including background knowledge. In this theory, domain ontology concepts and their properties are expressed by axioms. Each web service is expressed by a set of axioms that describes the service. The query (request) is expressed as a theorem in the language of application domain theory. Usually, the theorem expresses the existence of the entity (e.g. a citizenship, a name, etc) that the query is asking to find by given entity or entities. Using an automated theorem prover, it is proved that the theorem follows from the axioms of the application domain theory. The proof should be sufficiently constructive, so that from proving the existence of a desired entity we can construct a program of finding it.

As the automated theorem prover we use FOL theorem prover Gandalf [4] extended for synthesis of programs in order to be suitable for web service composition tasks and called RqIGandalf as in [6]. The core method employed by the system is the classical combination of the resolution method with the ans-predicate mechanism; see [13, 15]. The system is looking for suitable instantiations of the variable(s) in the query. These variables will be the arguments of the special ans-predicate. A clause containing only the ans-predicates is considered to be an empty clause, with the arguments of the ans predicate representing the solution to the query.

The simplistic usage of the ans-predicate will not be sufficient for program synthesis. We will need additional rules and mechanisms. The four main extensions built into the core prover are as follows:

Built-in strings, dates and arithmetic.

- A PROLOG-like mechanism for deriving different alternative answers.
- A derivation rule for eagerly disambiguating clauses containing several answer literals.
- A derivation rule for converting logical conditions into term-level conditions. For example, if we have a clause  $\neg A \mid \text{ans}(s)$ , then we can derive  $\text{ans}(\text{if}(A, s))$  as a new clause, provided that  $A$  is computable. Resolution proof search extended by such rules may give us a result in the form of a term. Next, this term should be compiled to an executable program.

Our compilation stage consists of two phases as follows. First step is the loop-lifting conversion modifying the term. Second step constitutes compilation of the modified term to an executable Python function. We have decided to use Python for practical usability, and we are also avoiding comprehension and lambda-terms in the Python program.

An important methodological decision we have taken is to avoid any forms of explicit induction. We describe lists using special function symbols, which will eventually be translated to loop constructions. Similarly, we avoid error handling in logic. The synthesized programs will handle possible error situations instead.

### 3.2. Methodology

Goal of the system is to automatically find a plan for service composition as an answer to the user request.

The most important steps of the web service composition are as follows:

1. Giving domain ontology as well as descriptions of web services in the ELM language.
2. Performing reasoning tasks and composition of web services by program synthesis using the theorem prover RqlGandalf [6] that is extended version of Gandalf [4].
3. Extracting the result of program synthesis as a Python program corresponding to the required composite service.
4. Developing the composite service by refining and further implementing the synthesized Python program.

Given web services domain ontology and web services descriptions in ELM language, the logic-based component of our framework translates these to internal logic language of FOL theorem prover RqlGandalf (i.e. to extended FOL). The resulting theory is passed to RqlGandalf, where it can be used for reasoning tasks and for composition of web services by program synthesis as discussed in the beginning of this section. We now show the web service composition on the basis of a small example.

### 3.3. Examples

The first example describes a single service called "addresses" which will take a person code as input and will return a list of addresses for the corresponding person. One of the data fields in an address indicates a city. The query asks for a subset of towns the person has addresses in: all the towns with a name lexicographically less than "London".

Ontology and a web service description are as follows:

```
?C is personcode of skolem1(?C).
if ?C is city of ?X then ?C is town of ?X.
if ?C is personcode of ?P then
foreach(L,getfield(city,L),call(addresses,?C)) is city of ?P.
```

The query for service composition is as follows:

```
find ?T where 123456789 is personcode of ?P
and ?T is town of ?P and ?T<"London"
```

Resulting program in Python language is as follows:

```
def query():
    result=[]
    data=call("addresses",123456789)
    for i in data:
        if lessthan(getfield("city",i),"London"):
            result=result+[getfield("city",i)]
    return result
```

The second example adds another service called "towndata" which will take a name of a city as input and will return a structure containing information about the city. We will only use (and describe) the "county" field. The synthesized program will have to use both services, one of these in a loop.

Ontology and a web service description is as follows:

```
?C is personcode of skolem1(?C).
if ?C is city of ?X then ?C is town of ?X.
if ?C is personcode of ?P then
foreach(L,getfield(city,L),call(addresses,?C)) is city of ?P.
if ?T is town of ?P then
getfield(county,call(towndata,?T)) is county of ?X.
```

Query is presented as follows:

```
find ?C where 123456789 is personcode of ?P
and ?C is county of ?P.
```

Resulting synthesised program in Python looks like the following:

```
def query():
    result=[]
    data=call("addresses",123456789)
    for i in data:
        result=result+\\
        [getfield("county",call("towndata",getfield("city",i)))]
    return result
```

A proof of concept implementation of the web service composition framework is done in the field of state information systems [6].

#### 4. Related Work

Using rule-based languages for developing methodologies for creation of database applications is not a new idea in the database research community. Active databases [16, 17] and deductive databases [18, 19, 20] are the most important research fields in this respect. In contrast to database-oriented approaches, our approach is rule-oriented

meaning that it offers declarative rule language and corresponding rule solver separately from database management system.

Well-known logic approaches applied to rule systems have been concentrating around using Prolog [21], Datalog [22, 23] or Description Logics [24].

Regarding to automation of web service composition, the approaches proposed fall into two main categories based on workflow models or on AI planning. Our work is tightly related to AI planning approaches based on automated program synthesis that relies on strong theorem proving technology. In [25, 26] available services and user requirements are described in FOL, and then constructive proofs are generated with SNARK theorem prover. Nevertheless, our approach differs from their approach in that they do not consider web services. We also enhanced FOL theorem prover in order to meet requirements of synthesis of web services.

Other logical approaches are used for web service composition in [27, 28, 29]. In [27] linear logic (LL) is used. For external presentation of web services they use semantic web service language DAML-S and for composition process they translate web services into extralogical axioms and proofs in LL. Service composition tool SWORD [28] generates composite service plans by using rule-based plan generation implemented in Java. [29] have developed semi-automatic service composition prototype, which consists of two basic components: a composer and an inference engine. First user selects the service he/she is interested in, and then inference engine finds all the other services that can supply appropriate data for selected service input.

## 5. Conclusion

We have provided a new conceptual and technological framework for using a rule language and rule engine for capturing application semantics in modern web-based systems. The approach is centered on the integration of a rule system with relational database systems and user interfaces using the middleware server Xstone. The rule system enables to deal with two aspects of semantics in web-based systems: business rules and web service composition logic.

The proposed framework and technology are implemented for Linux platform as open source software available at [6].

## Acknowledgements

This work was partially funded by Enterprise Estonia funding within R&D project RQL “Rule-based databases for creation of web services” and ESF grant 5766.

## References

- [1] Visual Rule Studio – the rule based expert system for Visual Basic .NET developers. RuleMachines, Web site [cited 2006 October 18]. Available from: <http://www.rulemachines.com/>.
- [2] The ARTEnterprise – a suite of rules-based, decisioning software components. MindBox, Web site [cited 2006 October 18]. Available from: <http://www.mindbox.com>.
- [3] ILOG’s business rule management system. ILOG, Web site [cited 2006 October 18]. Available from: <http://www.ilog.com/>.
- [4] T. Tammet, Gandalf. *Journal of Automated Reasoning*, **18** (2), (1997), 199-204.

- [5] Data Exchange Layer X-Road. X-Road, Web site [cited 2006 October 18]. Available from: <http://www.ria.ee/27309>.
- [6] Xstone/Rql – a suite of middleware/web service-style server tools for building complex software systems. Rql, Web site [cited 2006 October 18]. Available from: <http://www.ttu.ee/it/xstone>.
- [7] T. Tammet, H-M. Haav, V. Kadarpiik, M. Kääramees, A rule-based approach to web-based application development. O. Vacilecas et al (Eds), *Proceedings of the 2006 Seventh International Baltic Conference on Databases and Information Systems*, IEEE, (2006), 202-211.
- [8] H-M. Haav, T. Tammet, V. Kadarpiik, K. Kindel, M. Kääramees, A semantic-based Web service composition framework. *Information System Development*, Proceedings of 15th International Conference on Information Systems Development (ISD 2006), Springer-Verlag, New-York, (2007) (to appear).
- [9] T. Tammet, Extended Logical Markup Language (ELM) model and syntax specification. 2004 [cited 2006 October 18]. Available from: <http://deepthought.ttu.ee/it/elm/ecl.html>.
- [10] T. Tammet, Chain resolution for the Semantic Web. *Proceedings of Second International Joint Conference on Automated Reasoning (IJCAR 2004)*, LNCS 3097, Springer Verlag, (2004), 307-320.
- [11] T. Tammet, Extended Common Logic (ECL) model and syntax specification. 2004 [cited 2006 October 18]. Available from: <http://deepthought.ttu.ee/it/elm/ecl.html>.
- [12] T. Bernes-Lee, D. Conolly, S. Hawke, Semantic Web tutorial using N3. 2003 [cited 2006 October 18]. Available from: <http://www.w3.org/>.
- [13] C. Green, Application of theorem-proving for problem solving. *Proceedings of 1st International Joint Conference on Artificial Intelligence*, (1969), 219-239.
- [14] Z. Manna and R. Waldinger, A deductive approach to program synthesis. *ACM Transactions on Programming, Languages, and Systems*, 2, (1980), 90-121.
- [15] T. Tammet, Completeness of resolution for definite answers. *Journal of Logic and Computation*, 5(4), (1995), 449-471.
- [16] M. Morgenstern, Active databases as a paradigm for enhanced computing environments. M. Schkolnick and C. Thanos (Eds), *Proceedings of the 9th VLDB Conference*, (1983), 34-42.
- [17] J. Widom and S. Ceri, *Active Database Systems - Triggers and Rules For Advanced Database Processing*. Morgan Kaufmann, San Francisco, 1996.
- [18] J. Minker (Ed), *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Publishers, Inc., 1988.
- [19] I. Petrounias and P. Loucopoulos, A rule-based approach for the design and implementation of information systems. M. Jarke (Ed), *Proceedings of EDBT '94*, Springer-Verlag, Cambridge, U.K, (1994), 159-172.
- [20] M. Matskin and H-M. Haav, A deductive object-oriented approach to information system modeling. J. Eder and L. A. Kalinichenko (Eds), *Advances in Databases and Information Systems*, Workshops in Computing Series, Springer, (1996), 459-479.
- [21] L. Sterling and E. Shapiro, *The Art of Prolog*. MIT Press, 1994.
- [22] J. Ullman, *Database and Knowledge Base Systems*, 1-2. Computer Science Press, 1989.
- [23] C. Zaniolo, S. Tsur and H. Wang, LDL++ documentation and Web demo. 2002 [cited 2006 October 18]. Available from: <http://www.cs.ucla.edu/ldl>.
- [24] F. Baader et al (Eds), *The Description Logic Handbook*. Cambridge University Press, 2003.
- [25] R. Waldinger, Web agents cooperating deductively. J. L. Rash, et al (Eds), *Proceedings of First International Workshop on Formal Approaches to Agent-Based Systems (FAABS 2000)*, LNAI 1871, Springer-Verlag, (2001), 250-262.
- [26] R. Waldinger and J. Shrager, Deductive discovery and composition of resources. *Proceedings of RoW2006: Reasoning on the Web*, Edinburgh, Scotland, (2006).
- [27] J. Rao, P. Küngas, M. Matskin, Logic-based Web services composition: from service description to process model. *Proceedings of the 2004 IEEE International Conference on Web Services (ICWS'2004)*, San Diego, California, USA, IEEE Computer Society Press, (2004), 446-453.
- [28] S. R. Ponnekanti, A. Fox, SWORD: A developer toolkit for Web service composition. *Proceedings of Eleventh World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, (2002).
- [29] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau, HTN planning for Web service composition using SHOP2. *Journal of Web Semantics*, 1(4), (2004), 377-396.

This page intentionally left blank



# Application of AI Methods

This page intentionally left blank

# A Concept Map Based Intelligent System for Adaptive Knowledge Assessment

Alla ANOHINA<sup>1</sup> and Janis GRUNDSPENKIS<sup>2</sup>

*Department of System Theory and Design, Riga Technical University  
Riga, Latvia*

**Abstract.** The paper describes a concept map based multiagent system that has been developed for learners' knowledge assessment and self-assessment in process oriented learning. The architecture of the system in terms of modules, their functions and interaction is presented. The special attention is given to the intelligent assessment agent, which at the moment is composed of communication, knowledge evaluation, interaction registering, and expert agents. The paper also discusses a novel approach to adaptive knowledge assessment using concept maps.

**Keywords.** Computer-assisted assessment, intelligent agents, adaptive assessment

## Introduction

During recent decades we are eye-witnesses of an evolution from the industrial age to the information age. This evolution causes the rise of the information society in its diverse reality. The foundation of this society is "informationalism", which means that defining activities in all realms of human practice are based on information technology organized in information networks and captured around information processing [1]. Modern organizations realize that knowledge is their most important asset. In [2] it is noted that "knowledge has become increasingly relevant for organizations since the shift from an industrial economy based on assembly lines and hierarchical control to a global, decentralized, cooperative, innovative and information-driven economy". As a consequence, new terms such as "knowledge work" and "knowledge society" emerge. The essence of knowledge work is turning of information into knowledge for the purposes of problem solving and decision making.

In this context it is worth to stress that new requirements, technologies and demands for highly skilled and educated workforce influence a whole society of every country that is moving towards the knowledge society. It is obvious that teaching and learning processes also should be changed for the purpose of more effective turning of information into knowledge. The quality of knowledge, which a person acquires within an educational institution depends on the quality of teaching and learning process which, in turn, has strong connection with knowledge assessment.

---

<sup>1</sup> Corresponding Authors: Alla Anohina, Riga Technical University, Kalku street 1, Riga, Latvia, LV-1658; E-mail: [alla.anohina@cs.rtu.lv](mailto:alla.anohina@cs.rtu.lv).

<sup>2</sup> Janis Grundspenkis, Riga Technical University, Kalku street 1, Riga, Latvia, LV-1658; E-mail: [janis.grundspenkis@cs.rtu.lv](mailto:janis.grundspenkis@cs.rtu.lv).

The paper presents an intelligent system which gives a teacher an opportunity to assess learners' knowledge level at each stage of the learning process, and to use assessment results for analysis of quality and suitability of learning material, for changing teaching methods timely, and promoting a learning course towards achievement of desirable characteristics of learners' knowledge. At the same time the system can be used as a tool for knowledge self-assessment from the learners' side in order to control their acquired knowledge level and to perform activities directed towards filling blanks in their knowledge. Moreover, the further enhancement of the system in the direction of adaptive knowledge assessment based on concept maps is discussed.

The remainder of this paper is organized as follows. Section 1 briefly discusses computer-assisted assessment and its problems. The concepts, which underlie the developed system, are described in Section 2. The architecture of the system from the point of view of modules and their functions is presented in Section 3. Section 4 introduces an intelligent agent, which comprises the core of the developed system. Section 5 gives an overview of system's testing results in different learning courses. The proposed approaches to concept map based adaptive knowledge assessment are described in Section 6. Section 7 discusses some related works. Finally, Section 8 presents conclusions and outlines directions for future work.

## **1. Computer-Assisted Knowledge Assessment**

According to [3] the term "computer-assisted assessment" refers to the use of computers in assessment, encompassing delivering, marking and analysis of assignments or examinations, as well as collation and analysis of data gathered from optical mark readers. The most widespread computer-assisted assessment systems are ones based on objective tests [4, 5] that offer a learner a set of questions, answers on which are pre-defined [3]. The mostly used question types are multiple choice questions, multiple response questions, graphical hotspot questions, fill in blanks, etc.

Computer-assisted assessment typically is included in virtual learning environments, e.g. Blackboard (<http://www.blackboard.com>) or WebCT (<http://www.webct.com>), or it can be implemented in the form of a specialized assessment system. In the last case there is a set of available tools both from institutional and commercial developers [5]. CASTLE (<http://www.le.ac.uk/castle>), and TAL (<http://www.tal.bris.ac.uk>) are examples of software developed within the framework of the institutional projects. Commercial tools are Hot Potatoes (<http://hotpot.uvic.ca>), Questionmark™ Perception™ (<http://www.questionmark.com/us/home.htm>), Respondus (<http://www.respondus.com>), and others. Analysis of these products allows to define the main functional capabilities of computer-assisted assessment systems: full functionality related to management of questions (creation, removing, editing, etc.), planning and running of knowledge assessment activities, as well as reporting on the performance of both learners and questions.

Computer-assisted assessment provides a number of advantages [5, 6, 7, 8, 9]: greater flexibility regarding place and time of assessment, potential for providing assessments for large number of learners efficiently, instant feedback to learners, extensive feedback to teachers, reduced errors in comparison with human marking, decreased time needed for supervising and marking of assessments, and potential for frequent assessments. Besides the advantages computer-assisted assessment systems

have also drawbacks [5, 6, 7, 9]: some types of questions cannot be marked automatically as computer-assisted assessment is suited to those questions which require a limited response, unsupervised computer-assisted assessment sessions present a risk of plagiarism and illegal use of other materials, and some learners may have poor skills of information technologies usage.

However, the main drawback of such systems is a level of intellectual behavior, which can be assessed. According to [4, 8], it is not above the fourth level in the well-known Bloom's taxonomy [10], which includes three levels of lower order skills (Knowledge, Comprehension, and Application), and three levels of higher order skills (Analysis, Synthesis, and Evaluation). However, in [3] this assertion is called to be erroneous, but it is pointed out that designing test questions to assess higher order skills can be time consuming and requires skill and creativity.

Only a few computer-assisted assessment systems, which assess higher levels of intellectual abilities and skills, have been developed. They are based on strongly subject and language dependent tasks such as essays or free-text responses [11, 12, 13, 14]. These systems use methods of natural language processing and, therefore, are extremely complicated regarding their structure and functional mechanisms.

Assessment of the learner's knowledge level is an essential function of intelligent tutoring systems [15, 16] which use methods and principles of artificial intelligence in their architecture and operation in order to provide the most suitable learning for learner's abilities, knowledge, characteristics and needs. In these systems assessment as a rule takes the form of tests or sequences of tasks after the learner has finished studying a particular topic, as well as it is also implemented providing the practical problem solving mode. However, analysis of available publications reveals that such issue as continuous assessment of learner's knowledge and skills has not met enough attention from developers of intelligent tutoring systems.

So, there is a need for support of systematic knowledge assessment and reasonable balance between requirements to assess higher levels of knowledge and complexity of an assessment system. We have developed a concept map based assessment system, which meets all mentioned requirements.

## **2. Underlying Concepts of the System**

Three concepts underlie the developed system: process oriented learning, concept maps and adaptive knowledge assessment. Let's consider each of them.

Qualitative teaching and learning process is characterized by the fact that it is based on strengths of a learner and compensates his/her weaknesses. It may be achieved by systematic assessment of a learner's knowledge level and the use of results for changing teaching methods and learning content timely in order to achieve desirable knowledge characteristics. Thus, assessment focuses not only on the final result, but on the process of knowledge acquisition and becomes its integral part promoting process oriented learning.

In process oriented learning a teacher divides a learning course into some stages. The notion of a stage is not strictly defined and it can be any logically complete part of a learning course, for example, a chapter or a topic. At the end of each stage the teacher makes assessment of a learner's knowledge level. Methods of assessment depend on the teacher and specificity of the learning course. Assessment in the proposed system is based on the notion of concept maps.

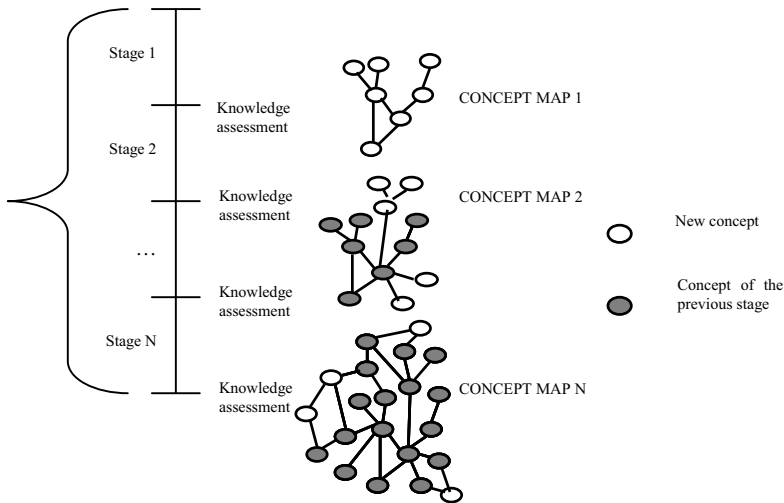
Concept mapping is an approach, which can be used “to externalize and make explicit the conceptual knowledge that student holds in a knowledge domain” [17]. Concept maps are a specific kind of a mental model and a method for representation and measuring of individual’s knowledge [18]. They are universal enough, as may be visualized as graphs using nodes and arcs that represent concepts and conceptual links, respectively. Conceptual links can be represented with or without linking phrases written on them. In the developed system two types of conceptual links are used. Important conceptual links show that relationships between the corresponding concepts are considered as important knowledge in the learning course. Less important conceptual links specify desirable knowledge. The linking phrases and direction are not used in the developed prototype. An example of a concept map for the learning course “SQL Fundamentals” is given in [19].

The first developed prototype of the system supports only one task: filling of a concept map structure. However, concept maps allow to provide learners with tasks with different level of difficultness taking into account degree of directedness [20]. Directedness is connected with information provided to learners. Tasks vary from high-directed to low-directed. High-directed concept map tasks provide learners with concepts, connecting lines, linking phrases, and a map structure. In contrast, in a low-directed concept map task learners are free to decide which concepts and how many of them should be included and how they will be related in their maps. In other words, tasks can be divided in a subset of “fill-in tasks” where learners are provided with a blank structure of a map and lists of concepts and linking phrases, and in a subset of “construct a map tasks” where learners are free to make their choices.

“Fill-in tasks” can be different, too. First, they vary on what is provided for learners (concept list, and/or linking phrases list), do they need to define something by themselves or do they need to use linking phrases at all. Second, the tasks vary on how a pre-defined concept map structure is provided: does it contains already some filled concepts and/or linking phrases, or it is empty. “Construct a map tasks” can have the same variety as “fill-in tasks” and also constraints on a number of concepts needed to use in the concept map, and on a structure (should it be strictly hierarchical or have some cycles).

Our proposed scheme of concept map usage in process-oriented learning is displayed in Figure 1. Thus, using the developed intelligent system a teacher prepares concept maps for each stage of the learning course, specifies one or several initial concepts and publication date of the map, and makes knowledge assessment at the end of each stage. It is important to stress that from one stage to another the initial concept map is extended on the basis of the following procedure. The teacher includes the concepts taught to learners at the first stage of the learning course and relationships between them into the first concept map of the learning course. At the second stage learners acquire new concepts. The teacher extends the initial concept map by adding new concepts, but doesn't change the relationships among already existing concepts. Thus, a concept map of each stage is not anything else than extension of a concept map of the previous stage. A concept map of the last stage displays all concepts in the learning course and relationships between them.

A variety of concept map based tasks allows to consider adaptive knowledge assessment. Computer adaptive knowledge assessment adapts tasks, questions or problems to a knowledge level of a particular learner. Therefore, learners with a low knowledge level do not receive very difficult tasks, but learners at a high achievement



**Figure 1.** The schema of concept map usage in process oriented learning

level are not required to solve too simple items. Adaptive assessment provides more accurate conclusions about the actual knowledge level of each learner [21].

### 3. System's Architecture and Operation

The developed intelligent system consists of an intelligent agent for assessment of learners' current knowledge level and a group of human agents, i.e. learners who are communicating with this agent.

The following scenario describes interaction between the system and its two users: a teacher and a learner. The teacher using the system creates concept maps for each stage of a learning course and defines their characteristics (initial concepts and publication date). During knowledge assessment the learner gets a structure of a concept map, which corresponds to the learning stage. At the first stage it is an empty structure with very few initial concepts defined by the teacher. In the subsequent stages new concepts are included in addition with those, which the learner has already correctly inserted during the previous stages. In both cases the set of concepts, which should be inserted into the structure of the concept map is given to the learner. After finishing the concept map, the learner confirms his/her solution and the intelligent assessment agent makes its analysis, comparing concept maps of the learner and the teacher on the basis of five patterns described in Section 4. The intelligent agent saves the final score of comparison and the learner's concept map in a knowledge base, and gives feedback to the learner about correctness of his/her solution. At any time the teacher has an opportunity to examine concept map completed by the learner and his/her score.

Modules of administrator, teacher and learner make the system's architecture. Their names display a category of system's users for which the module provides a set of functions. The modules interact sharing a common database which stores data about teachers and their learning courses, learners and groups of learners, teacher-created and learner-completed concept maps, learners' final score and system's users (Figure 2).

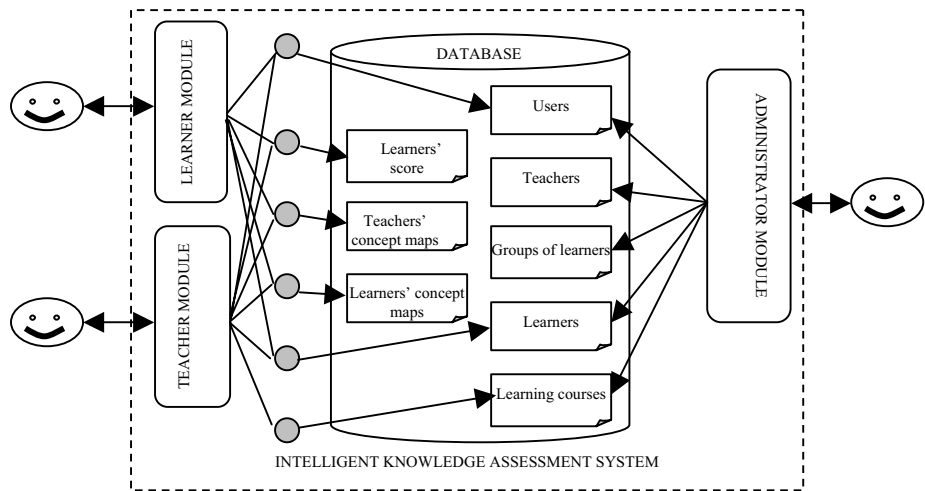


Figure 2. The architecture of the system

The administrator maintains the system and his/ her primary responsibilities are to manage data about users, learners and groups of learners, teachers and learning courses, using such functions of the administrator module as data input, editing, and deleting.

The teacher's module supports the teacher in the development of concept maps and of examining of learners' final score. Its main functions are the following: automatic providing of information on learning courses taught by the teacher and on learners studying a particular learning course, automatic providing of information about maximum score, the publication status and date of a particular concept map within a chosen learning course, tools for developing, editing and deleting concept maps, and tools for examining of learner-completed concept maps and score, as well as for deleting the results.

Graphical user interface is used for concept map development and editing. The teacher draws on a working surface using tools for concept inserting and linking them by two types of links (Section 2). When the teacher creates the first concept map in a particular learning course, he/she can freely add and delete concepts and links. However, if he/she creates a concept map of the next stage, he/she can freely operate only with elements of already not published maps of the previous stages.

Two kinds of information on learners' results are available: a learner-completed concept map with mistakes and incorrect places displayed on it, and a text pointing out learner's data, learning stage, and final score comparing with maximum score.

The learner's module includes the following functionality: automatic providing of information on learning courses studied by the learner, and on concept maps within learning courses (stage, status of publication, and learner's score), tools for completion of concept maps provided by the teacher, and tools for viewing the feedback after the learner has submitted his/her solution.

Feedback given to the learner comprises information on concepts uninserted into a map, incorrectly connected concepts, and final score. In the current version of the system the learner does not have an opportunity to see a correct concept map or to perform a task once again.



The system has been developed using the following tools: Borland JBuilder 9.0., JGraph, PostgreSQL DBMS 8.0.3. and JDBC drivers for PostgreSQL. The architecture of the system from the point of view of used technologies is given in [22].

#### 4. An Intelligent Assessment Agent

The intelligent agent who makes assessment of the learner's current knowledge level is a core of the system's intelligence and the basis of the learner's module. The developed prototype of this agent at the moment consists from four agents. The communication agent perceives the learner's actions on the working surface, i.e. concepts inserting into and removing from the structure of a concept map, and clicking on the buttons of solution submission and window closing. It is also responsible for visualization of a structure of a concept map, which is received from the agent-expert, and for the output of feedback coming from the knowledge evaluation agent. After the learner has confirmed his/her solution, the communication agent delivers the learner-completed concept map to the knowledge evaluation agent. This agent compares the concept maps of the learner and the teacher on the basis of recognition of five patterns of learner's solutions described below, and generates a feedback, which is delivered back to the communication agent. The interaction registering agent receives the learner-completed concept map from the communication agent and results of its comparison with the teacher-created concept map from the agent of knowledge evaluation, and stores them in a database. The agent-expert forms a structure of a concept map of the current learning stage on the basis of the teacher-created concept map and learner's concept map of the previous stage. The formed structure is delivered to the communication agent for its visualization on the working surface. The agent-expert also delivers a teacher-created concept map to the agent of knowledge evaluation for its comparison with a learner-completed concept map. Figure 3 displays the scenario of the system's operation described in Section 3 and the architecture of the intelligent assessment agent.

The knowledge evaluation agent is capable to recognize five patterns of learner solutions. It is based on assumption, that the fact that the learner understands presence of relationships between concepts has the primary value, while the type of link and the place of concepts within the general structure of a concept map are secondary things. Thus, the learner solutions, which the agent is capable to distinguish, are the following (Figure 4):

**Pattern 1.** The learner has related concepts as they are connected within a standard map of the teacher. In this case the learner receives 5 points regarding every important link and 2 points for less important link. Figure 4b shows that the learner has related concepts A and E, which fully match the same relationship in the teacher-created concept map (Figure 4a).

**Pattern 2.** The learner has defined a relationship, which does not exist in a concept map of the teacher. In this case he/she does not receive any points. In Figure 4c it is shown that the learner has related concepts A and H, but such relationship does not exist in the teacher-created map (Figure 4a).

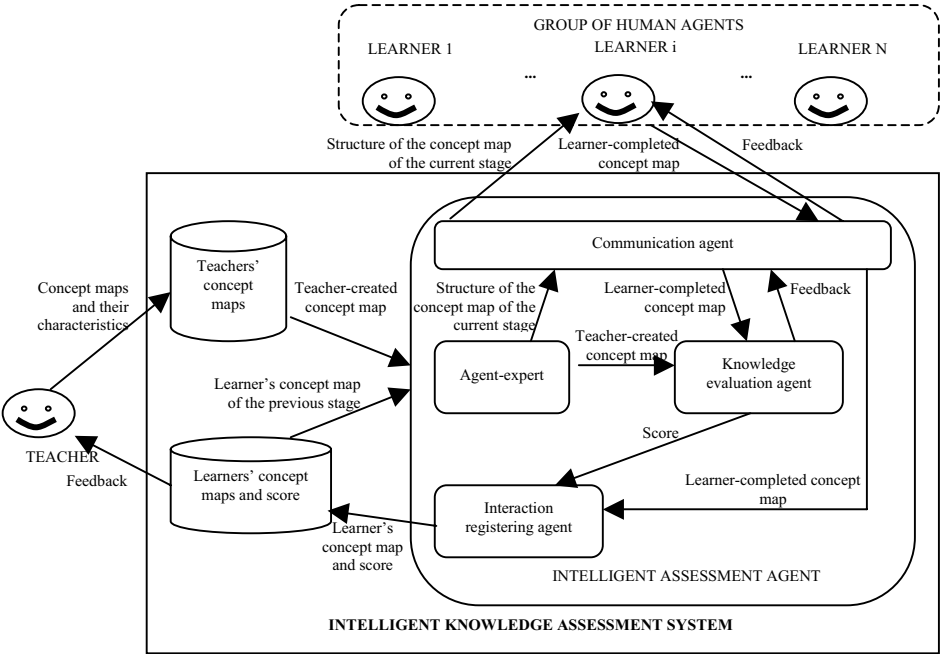


Figure 3. The architecture of the intelligent assessment agent

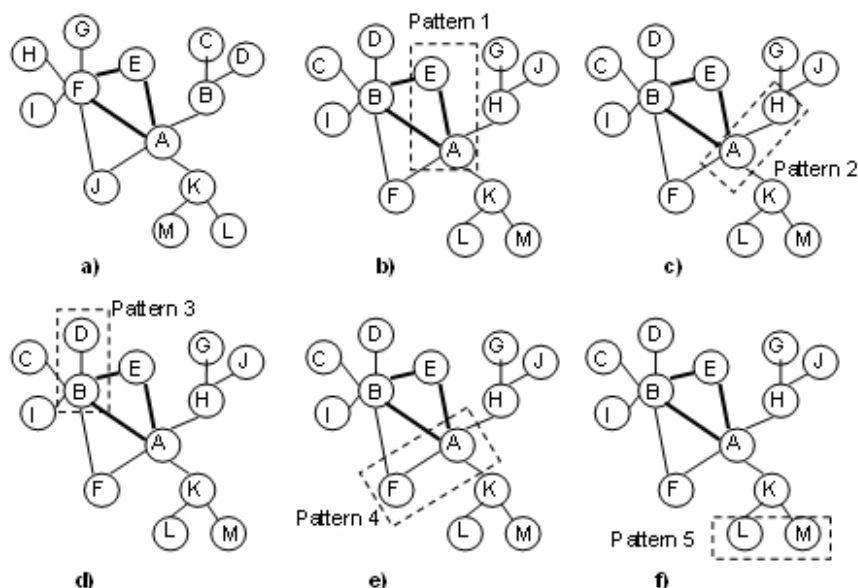
**Pattern 3.** The learner’s defined relationship exists in a standard map, the type of a link is correct, but at least one of concepts is placed in an incorrect place. The learner receives 80% from maximum score for that link. Figure 4d shows that the learner has defined relationship between concepts B and D, which also exists in the teacher’s map (Figure 4a). Both concepts are placed in the incorrect places although the type of the link is correct.

**Pattern 4.** The learner’s defined relationship exists in a standard map, the type of a link is wrong, and at least one of concepts is placed in an incorrect place. The learner receives 50% from maximum score for the correct link. This pattern is displayed in Figure 4e. Comparing the learner defined relationship between A and F with teacher’s one (Figure 4a) it is easy to see that concept F is placed in an incorrect place, as well as type of the link between concepts is less important instead of important link.

**Pattern 5.** A concept is placed in a wrong place, but its place is not important. The learner receives maximum score for a corresponding link. Figure 4f displays that the learner has exchanged concepts M and L by places comparing with the teacher-created concept map (Figure 4a).

5. System Testing Results

The operation of the developed system has been tested in six learning courses of different types (both engineering and social sciences). In the autumn of 2005 the testing took place on the learning courses “Information System Analyses and Development” and “Modelling and Formal Specification” in Vidzeme University College (Latvia), as



**Figure 4.** Patterns for learner's solution evaluation with the intelligent assessment agent: a) a teacher-created concept map; b) – f) patterns within a learner-created concept map

well as on the learning courses "Systems Theory Methods" and "The System of Teaching Methods" in Riga Technical University (Latvia). The testing on the learning courses "Fundamentals of Artificial Intelligence" and "Introduction to Artificial Intelligence" passed in spring of 2006 in Riga Technical University. One hundred and two students have been involved in the testing process. After testing students were asked to complete a questionnaire. Each questionnaire had fifteen questions, seven of them were devoted to the evaluation of system's performance and eight questions were related to the used approach based on concept maps. As a result eighty four questionnaires have been processed. The testing statistics is displayed in Table 1.

The example of system's operation is described in [22]. The students positively evaluated the chosen approach to knowledge assessment, as well as functionality and user interface of the system (questions and student answers are given in [22, 23]). They also stated desire to use such assessment technique in courses that will follow.

## 6. Enhancement of the System towards Adaptive Knowledge Assessment

The first prototype of the system presents all learners with the same structure of a concept map and initial concepts irrespective of a level of achievements of a particular learner. In order to improve functionality of the developed system and to make it more flexible we decided to exploit an idea of computer adaptive assessment.

**Table 1.** Testing statistics

Time	Learning course	Number of students involved in testing	Number of questionnaires received
Autumn 2005	Information System Analyses and Development	14	12
	Modeling and Formal Specification	33	26
	Systems Theory Methods	17	15
	The System of Teaching Methods	10	10
Spring 2006	Fundamentals of Artificial Intelligence	19	15
	Introduction to Artificial Intelligence	9	6
Totally		102	84


There are two time moments when the degree of task difficultness can be changed. Adaptation during the task performing is based on the voluntary request from the learner to decrease a level of difficulty of the task. For this purpose it is necessary to provide a tool (for example, a button) for the communication of the request. Adaptation after the finishing of the task is related with the decrease of task difficultness degree on the basis of the analysis of learner's submitted solution.

Generally, we propose two approaches to concept map based adaptive knowledge assessment: to change the number of empty places which the learner should fill performing the task and to change the type of the task.

The first approach keeps the same task: filling of a concept map structure. Performing the tasks the learner can directly request to decrease a difficulty level of the task. In this case the system will fill several empty places with correct concepts. This process continues until only a pre-defined number of empty places will remain or the learner will complete the task. So, at the first assessment stage the learner will receive an empty structure of a concept map with a very few initial concepts defined by the teacher. At the subsequent stages the structure will be extended due to the new concepts and relationships and will contain not only teacher defined concepts and concepts correctly inserted by the learner in the previous stage, but also concepts which were used to fill empty places decreasing the difficultness degree in the previous stage.

The second approach assumes enrichment of the system by tasks of various types. We have chosen 5 tasks which are ranged from the easiest to the most difficult based on the information given to the learner and the amount of job needed to be performed by the learner (Table 2). There are both “fill-in” and “construct a map” tasks. At the first assessment stage all learners receive the task of the average difficulty (the third tasks). The student can directly request to decrease a difficulty level of the task performing the task. The difficulty level of the subsequent stages depends on the level of the previous stage. If the student working at some level of difficulty reaches a minimum number of points, in the next stage the level of difficulty is increased. Otherwise, the level of difficulty remains the same.

**Table 2.** Tasks for concept map based adaptive knowledge assessment

The type of task	No.	The structure of a concept map	Linking phrases	Concepts	Difficultness degree	Comments
Fill-in	1	Is given	Inserted in the structure	Need to be inserted	 <p>The easiest</p>	Linking phrases that are inserted in the structure can help to find where a particular concept could be inserted
	2	Is given	Are not used	Need to be inserted		There is not any information which would allow to understand, where a particular concept should be inserted
	3	Is given	Need to be inserted	Need to be inserted		There is not any information which would allow to understand, where a particular concept should be inserted, as well as the volume of work is increased
Construct a map	4	Is not given	Are not used	Need to be related		
	5	Is not given	Need to be inserted	Need to be related		The volume of work is increased in comparison with Task 4
					The most difficult	

## 7. Related Works

The idea to computerized concept mapping is not new at all. A number of commercial and non-commercial graphical software packages and tools already exist, for example, AXON Idea Processor (<http://web.singnet.com.sg/~axon2000/>), Inspiration (<http://www.inspiration.com>), SMART Ideas™ (<http://www2.smarttech.com/st/en-US/Products/SMART+Ideas>), IHMC CmapTools (<http://cmap.ihmc.us>), and others, which allow to capture and visualize ideas and knowledge. These products provide such functions as concept map construction, navigation and sharing, and can be used as a useful learning tool, but they do not assess created concept maps.

One of the powerful concept map based assessment tool is COMPASS [24]. It is a Web-based system that provides assessment of the learners' knowledge level through various concept mapping tasks and supports the learning process generating the informative and tutoring feedback after the analysis of a learner's concept map.

The other example of assessment tool based on concept maps is described in [25]. It has two versions: one of them supports the task of filling in the blanks of incomplete structure of a concept map, other offers an opportunity to freely construct a concept map. Both versions provide evaluation and hint functions.

The developed system has two discriminative features in comparison to the mentioned tools. Both known systems consider assessment as a discrete event, while the system described in this paper supports process oriented learning and allows the teacher to extend the initially created concept map for the new stage of assessment. The second unique feature is an algorithm that compares the teacher's and learner's concept maps and is sensitive to the arrangement and coherence of concepts.

The idea of computer adaptive knowledge assessment is implemented mainly in the systems based on objective tests. It is supported by some software products such as Questionmark™ Perception™ (<http://www.questionmark.com/us/home.htm>) and TRIADS (<http://www.derby.ac.uk/assess/newdemo/mainmenu.html>). There is also some research in this area. PASS module [26] is based on adaptive testing and adaptive questions techniques, and can be integrated in an Adaptive Educational Hypermedia System in order to provide personalized assessment selecting the appropriate question for a learner according to the questions' parameters, the assessment parameters, and the current learner's knowledge level. E-TESTER [27] automatically creates questions based on e-learning content provided to a learner. The developed system in its turn is based on a novel approach regarding adaptive knowledge assessment using concept maps.

## 8. Conclusions and Future Work

The paper describes the concept map based multiagent system for learners' knowledge assessment at each stage of a learning course. The core of the system is the intelligent agent that is capable to recognize five patterns of learners' solutions. It is communicating with a group of human agents, i.e. learners, and at the moment consists from four agents: communication, knowledge evaluation, interaction registering and expert agents.

The developed system has some discriminative features in comparison with other concept mapping and computer-assisted assessment systems. Firstly, the system supports process oriented learning and allows the teacher to extend the initially created concept map for the new stage of assessment. The second unique feature is an algorithm that compares the teacher's and learner's concept maps and is sensitive to the arrangement and coherence of concepts. These two features have been already implemented. The third feature that is under development at the moment is concept map usage for adaptive knowledge assessment. The paper discusses two approaches to concept map based adaptive knowledge assessment and their implementation aspects.

The chosen approach to knowledge assessment, i.e. concept maps, can be successfully used for systematic knowledge assessment, as well as allow to assess higher order skills providing possibility to issue tasks with different level of difficultness and simultaneously do not require natural language processing.

The future work has three main directions. One of them is related with the deeper analysis of the proposed approaches to concept map based adaptive knowledge assessment. The next one focuses on implementation of both approaches in two different systems. The developed systems will be tested in learning courses of various types.

## Acknowledgment

This work has been partly supported by the European Social Fund within the National Program “Support for the carrying out doctoral study program’s and post-doctoral researches” project “Support for the development of doctoral studies at Riga Technical University”.

## References

- [1] M. Castells. *The Information Age: Economy, Society and Culture*. Malden, Mass: Blackwell, 2000.
- [2] U.M. Borghoff and R. Pareschi (eds.). *Information Technology for Knowledge Management*. Berlin: Springer, 1998.
- [3] Computer-assisted Assessment (CAA) Centre. Web site [cited 2006 September 10]. Available from: <http://www.caacentre.ac.uk>.
- [4] J. Bull. Introduction to computer-assisted assessment. [Cited 2006 September 10]. Available from: <http://asp2.wlv.ac.uk/celt/download.asp?fileid=44&detailsid=200008>.
- [5] Using computer assisted assessment to support student learning. The Social Policy and Social Work subject centre with the Higher Education Academy (SWAP). [Cited 2006 September 10]. Available from: <http://www.swap.ac.uk/elearning/develop6.asp>.
- [6] E-assessment. [Cited 2006 September 10]. Available from: <http://en.wikipedia.org/wiki/E-assessment>.
- [7] G. Lambert. What is computer aided assessment and how can I use it in my teaching? (Briefing paper), Canterbury Christ Church University College, 2004.
- [8] N. Moge and H. Watt. Chapter 10: The use of computers in the assessment of student learning. In: *Implementing Learning Technology*, G. Stoner (ed). Learning Technology Dissemination Initiative, 1996, pp. 50-57.
- [9] A. Oliver. Computer aided assessment - the pros and cons. [Cited 2006 September 10]. Available from: [http://www.herts.ac.uk/ltdu/learning/caa\\_procon.htm](http://www.herts.ac.uk/ltdu/learning/caa_procon.htm).
- [10] B. S. Bloom. *Taxonomy of educational objectives. Handbook I: The Cognitive Domain*. New York: David McKay Co Inc., 1956.
- [11] J. Burstein, C. Leacock and R. Swartz. Automated evaluation of essays and short answers. In: *Proceedings of the 5th International Computer Assisted Assessment Conference*, Loughborough University, UK, 2001, pp. 41-45.
- [12] C. Leacock and M. Chodorow. C-rater: scoring of short-answer questions. *Computers and the Humanities*, Vol. 37, No. 4, 2003, pp. 389-405.
- [13] J. Z. Sukkarieh, S. G. Pulman and N. Raikes. Auto-marking: using computational linguistics to score short, free text responses. In: *Proceedings of the 29th Conference of the International Association for Educational Assessment*, The Midland, UK, 2003.
- [14] D. Pérez, E. Alfonseca and P. Rodríguez. Application of the BLEU method for evaluating free-text answers in an e-learning environment. In: *Proceedings of the 4th International Language Resources and Evaluation Conference (LREC-2004)*, Lisbon, 2004, pp.1351-1354.
- [15] M. C. Polson and J. J. Richardson (eds.). *Foundations of Intelligent Tutoring Systems*. Hillsdale, N.J.: Erlbaum, 1988.
- [16] J. Beck, M. Stern and E. Haugsjaa. Applications of AI in Education. *ACM Crossroads*, Issue 3.1, 1996. [Cited 2006 September 10]. Available from: <http://www.acm.org/crossroads/xrds3-1/aied.html>.
- [17] A. J. Cañas, J. W. Coffee, M. J. Carnot, P. Feltovich, R. R. Hoffmann, J. Feltovich and J. D. Novak. A summary of literature pertaining to the use of concept mapping techniques and technologies for education and performance support. Report to the Chief of Naval Education and Training, Pensacola, Florida, IHMC, 2003.
- [18] D. T. Croasdel, L. A. Freeman and A. Urbaczewski. Concept maps for teaching and assessment. *Communications of the Association for Information Systems*, Vol. 12, 2003, pp. 396-405.
- [19] A. Anohina, G. Stale and D. Pozdnyakov. Intelligent system for student knowledge assessment. In: *Scientific Proceedings of Riga Technical University*, 5th series, Computer Science, Applied Computer Systems, Riga, 2006 (to appear).
- [20] M. A. Ruiz-Primo. Examining concept maps as an assessment tool. In: *Proceedings of the 1st Conference in Concept Mapping*, Pamplona, Spain, 2004.
- [21] E. Papanastasiou. Computer-adaptive testing in science education. In: *Proceedings of the 6th International Conference on Computer Based Learning in Science*, Nicosia, Cyprus, 2003, pp. 965-971.

- [22] A. Anohina and J. Grundspenkis. Prototype of multiagent knowledge assessment system for support of process oriented learning. In: Proceedings of the 7th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2006), Vilnius, Lithuania, 2006, pp. 211-219.
- [23] A. Anohina, V. Graudina and J. Grundspenkis. Intelligent system for learners' knowledge self-assessment and process oriented knowledge control based on concept maps and ontologies, Annual Proceedings of Vidzeme University College, Latvia, 2006 (to appear).
- [24] E. Gouli, A. Gogoulou, K. Papanikolaou and M. Grigoriadou. COMPASS: an adaptive Web-based concept map assessment tool. In: Proceedings of the 1st Conference in Concept Mapping, Pamplona, Spain, 2004.
- [25] K. E. Chang, Y. T. Sung and S. F. Chen. Learning through computer-based concept mapping with scaffolding aid. *Journal of Computer Assisted Learning*, Vol. 17, 2001, pp. 21-33.
- [26] E. Gouli, K. A. Papanikolaou and M. Grigoriadou. Personalizing assessment in adaptive educational hypermedia systems. In: Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Malaga, Spain, 2002, pp. 153 – 163.
- [27] C. Guetl, H. Dreher and R. Williams. E-TESTER: A computer-based tool for auto-generated question and answer assessment. In: Proceedings of the E-Learn 2005-World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, Vancouver, Canada, 2005, pp. 292-293.



# Semantic Interoperability between Functional Ontologies

Nacima MELLAL <sup>a,1</sup>, Richard DAPOIGNY <sup>a</sup> and Laurent FOULLOY <sup>a</sup>

<sup>a</sup> *Laboratoire d'Informatique, Systèmes, Traitement de l'Information et de la Connaissance LISTIC, ESIA, Université de Savoie, B.P.806, F-74016 ANNECY Cedex, France*

**Abstract.** The specification of distributed engineering systems, can be simplified with the introduction of the well-known service paradigm. Generally, in goal driven-approaches in which services are related to functional goals, the use of concepts through well-formed structures (i.e., ontologies) provide the appropriate functionalities. Nowadays, it is also essential to take into account the crucial aspects of the dynamic services, that is to say their ability to adapt and to be composed in order to complete their task. As a result, we observe a growing need for service composition (i.e., goal composition) together with emerging difficulties such as the selection of appropriate structures describing service functionalities or the problem of semantic heterogeneity resulting from ontology alignment in distributed environments. Therefore, this paper presents a well-founded mathematical approach in which services are expressed in the engineering framework with goal structures. The model builds on channel theory to enable semantic interoperability between functional ontologies. This results into a computationally tractable system in which functional compositionality across cooperative systems can occur in a sound way.

**Keywords.** Functional ontology, IF-based model, alignment, distributed systems, software agent, semantic interoperability

## Introduction and Motivation

The notion of service is central in the description and functioning of distributed systems. Based on a teleological assumption <sup>2</sup>, services are related to goals through the paradigm 'Service as global goal achievement'. From the distributed perspective, these systems suffer from the increasing complexity caused by the huge amount number of services available. This is related to a lack of global vision at design time. Therefore, to face up to these difficulties, the development of dynamic and automatic techniques for the composition and the fusion of services is required, reducing as well the implementation. Current solutions are limited and are primarily static and manual. Reasoning about the composition of services requires a representation and a description of these services with a functional model [1]. Ontologies have gained popularity as means for generating explicit formal vocabulary to share between applications. Moreover, they have proved to be a successful approach to provide appropriate tools for re-use and formalization [2].

---

<sup>1</sup>E-mail: nacima.mellal@univ-savoie.fr

<sup>2</sup>in which purposes are assigned to functions and mechanisms are described by function implementation

Recently, much attention has been paid on finding semantic correspondences between ontologies, this is known as ‘semantic interoperability’. Alignment between ontologies is a critical challenge for semantic interoperability. Efforts provided in this area focus on finding semantic correspondences between ontologies where major works are those of Kalfoglou and his co-authors in [3], they made some steps towards semantic integration by proposing a mathematically sound application of Information Flow model from Barwise and Seligman theory [4] to enable semantic interoperability of separate ontologies representing similar domains. N.Noy also studied this issue by giving a brief review of ontology-based approaches to semantic integration [5]. Last but not least, in [6] Doan and Halevy have focused on the semantic-integration work in the database community.

To give a formal characterization of functional ontology alignment, we propose a methodology based on a sound mathematical model IF Information Flow. The automated ontology alignment capabilities are specified by software agents which interoperate semantically to achieve a common purpose. The overall goal of the present paper will try to give answers to the following questions:

- What is the structure of the ontology and how is it related to other ontologies?
- What are the features of the IF based model and how it is able to automate the ontology alignment ?
- How do agents specify the Ontology Alignment and how is the communication achieved ?

In the present work, we focus on functional ontologies describing distributed and complex systems. The alignment of ontologies is used to discover the correspondences between functional ontologies. As a crucial topic, information exchange between functional hierarchies must occur in a semantically sound manner. In [7] ontology alignment has been defined as any formal description of the (semantic) relationship between ontologies. In [8], authors describe the process of alignment using IF theory, we have extended their work, to functional hierarchies where the classification relation expresses the functional dependency. Functional ontologies alignment is a process which describes how one or more goals of an ontology can be semantically mapped to goal(s) of other ontologies in a sound and automatic way, without altering the original ontologies. Such a process must not be considered as a simple pattern matching process, but rather an intentional alignment process since the resulting goal dependencies must respect the sum of the local logics both on the syntactic and the semantic level.

This article is divided into 5 mains sections. In section one, we give the definitions of main concepts in the IF theory. In section two, an application example is detailed in order to make clarification progressively as the different proposed definitions go on. In section three, the functional ontology is introduced through the description of goals (service) structure and the construction of its functional ontologies is explained. In section four, the ontology alignment process is described. Finally, in section five, we discuss the specification of ontology alignment by software agents.

## 1. IF-Based Theory

### 1.1. Local IF Structures

**Definition 1** An **IF Classification**  $A$  is a triple  $\langle tok(A), typ(A), \models_A \rangle$ , which consists of:

1. a set  $tok(A)$  of objects to be classified known as the instances or particulars of  $A$  that carry information,
2. a set  $typ(A)$  of objects used to classify the instances, the types of  $A$ ,
3. a binary classification relation  $\models_A$  between  $tok(A)$  and  $typ(A)$  that tells one which tokens are classified as being of which types.

The meaning of the notation  $a \models_A \alpha$  is "instance  $a$  is of type  $\alpha$  in  $A$ ". IF classifications are related through infomorphisms. Infomorphisms are the links between classifications.

**Definition 2** Let  $A$  and  $B$  be IF classifications. An **Infomorphism** denoted  $f = \langle f^\wedge, f^\vee \rangle : A \rightleftarrows B$  is a contravariant pair of functions  $f^\wedge : typ(A) \rightarrow typ(B)$  and  $f^\vee : tok(B) \rightarrow tok(A)$  which satisfies the fundamental property:  
 $f^\vee(b) \models_A \alpha$  iff  $b \models_B f^\wedge(\alpha)$  for each  $\alpha \in typ(A)$  and  $b \in tok(B)$

Regularities in a distributed system are expressed with IF theories and IF logics. In distributed system, IF-theories can be seen as an idealized version of the scientific laws supported by a given system.

**Definition 3** An **IF theory**  $T$  is a pair  $\langle typ(T), \vdash_T \rangle$  where  $typ(T)$  is a set of types and  $\vdash_T$ , a binary relation between subsets of  $typ(T)$ .

Let  $A$  be a classification. A token  $a \in tok(A)$  satisfies the constraint  $\Gamma \vdash \Delta$  where  $(\Gamma, \Delta)$  are subsets of  $typ(A)$ , if  $a$  is of some types in  $\Delta$  whenever  $a$  is of every type in  $\Gamma$ . If every token of  $A$  is constrained by  $(\Gamma, \Delta)$ , we have obviously  $\Gamma \vdash_A \Delta$  and  $\langle typ(A), \vdash_A \rangle$  is the theory generated by  $A$ . A theory  $T$  is said regular if for all  $\alpha \in typ(T)$  and for arbitrary subsets  $\Gamma, \Delta, \Gamma', \Delta', \Sigma'$  of  $typ(T)$ , the following properties hold:

- The Identity:  $\alpha \vdash_T \alpha$
- The Weakening: if  $\Gamma \vdash_T \Delta$  then  $\Gamma, \Gamma' \vdash_T \Delta, \Delta'$
- The Global cut: if  $\Gamma, \Gamma' \vdash_T \Delta, \Delta'$  for any partition<sup>3</sup>( $\Gamma', \Delta'$ ) of  $\Sigma'$ , then  $\Gamma \vdash_T \Delta$ , for all  $\Gamma, \Delta \vdash_T typ(T)$

**Definition 4** A **local logic**  $\mathcal{L} = \langle tok(\mathcal{L}), typ(\mathcal{L}), \models_{\mathcal{L}}, \vdash_{\mathcal{L}}, N_{\mathcal{L}} \rangle$  consists of a regular IF theory  $th(\mathcal{L}) = \langle typ(\mathcal{L}), \vdash_{\mathcal{L}} \rangle$ , an IF classification  $cla(\mathcal{L}) = \langle tok(\mathcal{L}), typ(\mathcal{L}), \models_{\mathcal{L}} \rangle$  and a subset  $N_{\mathcal{L}} \subseteq tok(\mathcal{L})$  of normal tokens which satisfy all the constraints of  $th(\mathcal{L})$ .

A token  $a \in tok(\mathcal{L})$  is constrained by  $th(\mathcal{L})$ . Given a constraint  $(\Gamma, \Delta)$  of  $th(\mathcal{L})$ , whenever  $a$  is of all types in  $\Gamma$ , then  $a$  is of some type in  $\Delta$ . An IF logic  $\mathcal{L}$  is sound if  $N_{\mathcal{L}} = tok(\mathcal{L})$ . In this paper, we restrict the classification relation to normal instances,

<sup>3</sup>A partition of  $\Sigma'$  is a pair  $(\Gamma', \Delta')$  of subsets  $\Sigma'$ , such that  $\Gamma' \cup \Delta' = \Sigma'$  and  $\Gamma' \cap \Delta' = \emptyset$

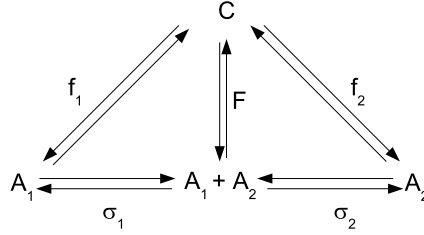


Figure 1. IF channel

limiting ourselves to sound logics. This assumption is required to enable ontology sharing or ontology matching [9,10]. In summary, each component of a distributed system is described with a sound local logic integrating a classification and its associated theory.

$$\mathcal{L} = \langle tok(\mathcal{L}), typ(\mathcal{L}), \models_{\mathcal{L}}, \vdash_{\mathcal{L}} \rangle \quad (1)$$

Let us introduce the distributed IF structures, which base on the IF channel. IF channel models the information flow between IF classifications. The local logic is the ‘what’ of IF, the channel is the ‘why’.

### 1.2. Distributed IF Structures

The IF channel models the information flow between IF classifications.

**Definition 5** An **IF channel** consists of two classifications  $A_1$  and  $A_2$  connected through a core classification  $C$  by means of two infomorphisms  $f_1$  and  $f_2$ .

Since local logics are inclusive concepts combining the concepts of classification and theory, they capture a more general knowledge than single classifications. Therefore there is a need to consider distributed IF logics of IF channels.

**Definition 6** Given a binary channel  $\mathcal{C} = \{f_1 : A_1 \rightleftharpoons C, f_2 : A_2 \rightleftharpoons C\}$  with a logic  $\mathcal{L}$  on the core classification  $C$ , the distributed logic  $DLog_{\mathcal{C}}(\mathcal{L})$  of  $\mathcal{C}$  generated by  $\mathcal{L}$  is such as:

$$DLog_{\mathcal{C}}(\mathcal{L}) = F^{-1}[\mathcal{L}] \quad (2)$$

The local logic on the sum  $A_1 + A_2$  which represents the reasoning about relations among the components is also referred as the distributed logic of  $\mathcal{C}$  generated by  $\mathcal{L}$  while  $F$  denotes the infomorphism  $F : (A_1 + A_2) \rightleftharpoons C$ .  $\sigma_1$  and  $\sigma_2$  are also infomorphisms, where,  $\sigma_1 : A_1 \rightleftharpoons (A_1 + A_2)$  and  $\sigma_2 : A_2 \rightleftharpoons (A_1 + A_2)$ , (see Figure 1).

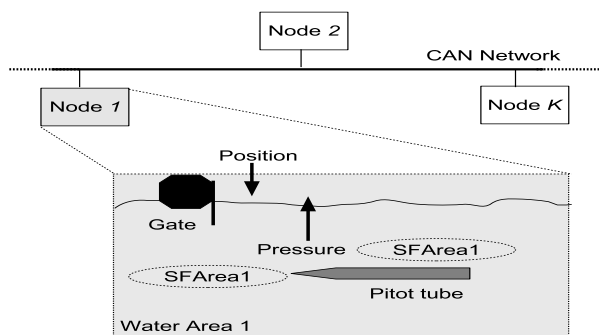


Figure 2. Hydraulic channel system

## 2. An Application Example

We propose a case study which concerns an open channel hydraulic system composed of nodes which represent subsystems (S1, S2, ...) connected with a CAN network. Each subsystem used in the open-channel irrigation channel is located near a water gate. Each subsystem performs two pressure measurements from a Pitot tube and is able to react accordingly and to modify the gate position with the help of a brushless motor. The Pitot tube allows two pressure measurements, a static pressure in spatial areas denoted SFArea, and a dynamic pressure measurement in areas denoted DFArea. The local area surrounding the two previous is respectively referred as WaterArea (see Figure 2.). The node is composed of two input sensors for pressure measurement and an actuator able to regulate the water level. In addition the node can communicate through a network with other similar systems.

## 3. Functional Ontologies

Basically, distributed systems are described by services, but composition of goals is more intuitive than composition of services. In addition, goals hide service complexity and offer the ability to be independent of possibly fluctuating services. Therefore, we map services to goals following in such away, the teleological assumption [11,12]. As a result, services (goals) are expressed by a functional model<sup>4</sup>. Integration of goal modeling facilitates service composition through goal hierarchies. In order to generalize and to enhance re-use in distributed environments, we introduce local goal ontologies. Ontologies are defined as an explicit specification of a conceptualization. In general, an ontology should satisfy an explicit definition of concepts and relations among them. We call **Functional Ontology** the ontology of functional concepts. In our context, the concepts have a functional nature which consist of goals to be achieved and the relations among them are the causal order of achievement. In this section of paper, we explain the concept of functional ontology and the relations between them by proposing formal definitions.

<sup>4</sup>We restrict the goal modeling to hard goals, that is functional goals.

### 3.1. Goals Are Concepts of the Functional Ontology

We propose the main concepts we use to define the concepts of functional ontology.

- **Physical Role *PR*** describes the physical quantity (property) related at least to a spatial location.
- **Physical Entity Type *PE-Type***: PE-Type is any spatial location.
- **Physical Entity Token *PE-Token***: PE-Token is a specified spatial location.
- **Physical Context Type *PC-Type***: is a tuple:  $\xi_i \stackrel{def}{=} (r, \{\psi_1, \psi_2, \dots, \psi_n\})$ . Where  $r$  is PR,  $\psi_1, \psi_2, \dots, \psi_n$  is a set of PE-Type. According to our example,  $(pressure, liquid\_volume)$  is a PC-Type, where  $liquid\_volume$  describes the PE-Type related to the PR  $pressure$ .
- **Physical Context Token *PC-Token***: Similar definition of PC-Types holds for PC-Tokens by replacing PE-Type with PE-Token. For example,  $(pressure, SF\ Area)$  is the PC-Token, where  $SF\ Area$  is the PE-Token.

#### Definition 7 Goal Type *G-type*

Given  $A$ , a non-empty set of action verbs,  $\Xi$ , a non-empty set of PC-Types, a G-Type denoted  $\gamma_i$  is a pair:  $\gamma_i \stackrel{def}{=} (\{a\}, \{\xi_1, \dots, \xi_k\})$ , where the singleton  $\{a\} \subseteq A$  and  $\{\xi_1, \dots, \xi_k\} \subseteq \Xi$ .

According to our example, the possible G-Types are:

- $\gamma_1 = (\{to\_acquire\}, \{(pressure, liquid\_volume)\})$
- $\gamma_2 = (\{to\_compute\}, \{(velocity, channel\_part)\})$
- $\gamma_3 = (\{to\_compute\}, \{(level, channel\_part)\})$
- $\gamma_4 = (\{to\_send\}, \{(velocity, channel\_part), (level, channel\_part)\})$
- $\gamma_5 = (\{to\_receive\}, \{(velocity, channel\_part), (level, channel\_part)\})$
- $\gamma_6 = (\{to\_compute\}, \{(level, \{channel\_part, channel\_part\})\})$
- $\gamma_7 = (\{to\_compute\}, \{(offset, actuator)\})$
- $\gamma_8 = (\{to\_receive\}, \{(offset, actuator)\})$
- $\gamma_9 = (\{to\_act\_upon\}, \{(position, actuator)\})$

- **Remark:** Similar definition holds for G-Tokens by replacing PC-Types with PC-Tokens

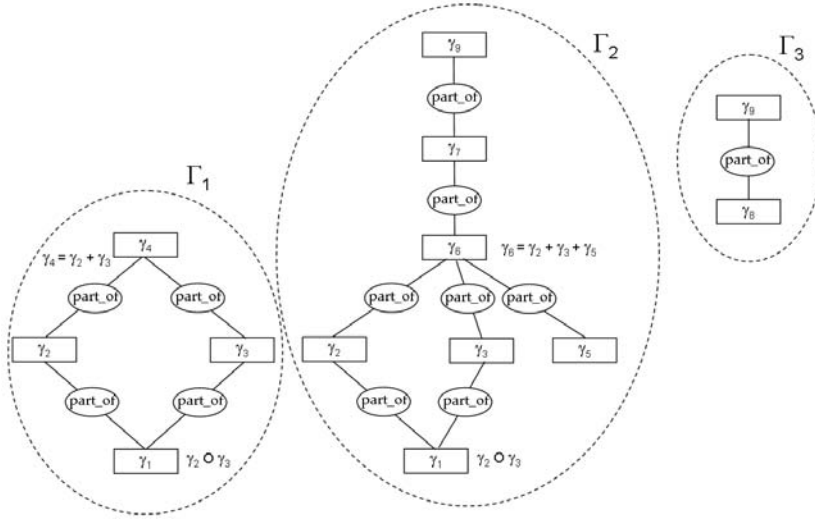
### 3.2. Relationships of the Functional Ontology

The ontology includes goal types as concepts and three core notions: the functional part-of ordering relation, the overlap between goals, and the sum (fusion) of goal types:

- $\sqsubseteq$  : The functional part-of ordering relation.
- $\circ$  : The overlap between goal types.
- $+$  : The sum (fusion) of goal types.
- "Functional Part-of relation  $\sqsubseteq$ "

A goal type  $\gamma_i$  influences functionally a goal type  $\gamma_j$  if the only way to achieve  $\gamma_j$  is to have already achieved  $\gamma_i$ , it will be denoted:

$$\gamma_i \sqsubseteq \gamma_j \tag{3}$$



**Figure 3.** Functional ontology for each service

- ‘Overlap O’ The overlap is referred by the binary relation  $O$  between goal types and specified with the following ontological axiom:

$$\gamma_i O \gamma_j \stackrel{def}{\equiv} \exists \gamma_k (\gamma_k \sqsubseteq \gamma_i \wedge \gamma_k \sqsubseteq \gamma_j) \quad (4)$$

- ‘Sum +’ The goal sum (fusion) is a n-ary relation between constituent goals such that the resulting goal overlaps all and only those goals that overlap at least one of the constituents:

$$\gamma_i + \gamma_j \stackrel{def}{\equiv} \exists \gamma_p \forall \gamma_k (\gamma_k O \gamma_p \Leftrightarrow (\gamma_k O \gamma_i \vee \gamma_k O \gamma_j)) \quad (5)$$

Several structures are used to represent concepts. In addition, the concept hierarchy is mostly in a tree. Since the goal types appear to be components of a hierarchical structure, it is worth describing them by means of a subsumption hierarchy [13] (i.e., a concept lattice) to represent relationships between G-Types.

**Definition 8** A functional part-of hierarchy  $\mathcal{F}$  is described by the following tuple:  $\mathcal{F} = (\Gamma, \sqsubseteq, O, +)$ , where  $\Gamma$  is a finite set of goal types,  $\sqsubseteq$  is a partial order on  $\Gamma$ ,  $O$  is the overlap relation, and  $+$ , the fusion relation on goal types. According to our example, we have three main functional part-of hierarchies (see Figure 3).

A constraint denoted  $\gamma_i \vdash \gamma_k$  represents the fact that a physical context type of  $\gamma_i$  is also a context of the goal  $\gamma_k$ . The following definitions relate goal overlaps and goal fusion to their Gentzen sequents <sup>5</sup>(the above definitions are extensible to more than two goals).

<sup>5</sup>The syntax of Gentzen sequent is interpreted by its symbol  $\vdash$  as an implication, the comma on the left is interpreted like a conjunction, the comma on the right like a disjunction.

- Example:  $\gamma_8 \vdash \gamma_9$ , this means that to act-upon PC-Type  $\xi_6 = (position, gate)$ , the offset referred by PC-Type  $\xi_5 = (offset, gate)$  should be received. So, there is a dependency between the PC-Type of  $\gamma_8 = (to\_receive, \xi_5)$  and  $\gamma_9 = (to\_act\_upon, \xi_6)$ .

#### 4. Ontology Alignment Process

We use the application example described above to illustrate our process of alignment. We constraint the example by two remote systems described by  $F_i^{(s1)}$  and  $F_j^{(s2)}$ , but our approach can be applied for more than two systems. We shall treat the problem of achieving a complex global goal  $\Gamma_2$  on system S1, that is, it must compare the water level in front of system S1 (in water area 1) with the level in front of the system S2 (in water area 2) in order to control the water level in the open-water channel. To realize this goal, the system needs to know whether it is achievable locally or if it needs some information from the remote system. This will be done by checking the hierarchy of the global goal and detecting if there will be some unavailable (invalid) physical context type at any goal type in this hierarchy. Considering the hierarchy  $\Gamma_2$ , the goal type  $\gamma_5$  requires to the physical contexts  $(velocity, ExtEnt)$  and  $(level, ExtEnt)$ , where *ExtEnt* refers to any spatial location.

The objective of the application is to demonstrate that the hierarchy of global goals in S1 ( $F_i^{(s1)}$ ), is able to select automatically the appropriate goal from the the hierarchy of global goals in system S2 ( $F_j^{(s2)}$ ) to complete its task. In terms of the channel-theoretic context, this means to know an IF theory that describes how the different types from  $F_i^{(s1)}$  and  $F_j^{(s2)}$  are logically related to each other, i.e., an IF theory on the union of types  $typ(A) \cup typ(B)$ . In such an IF theory a sequent like  $\alpha \vdash \beta$ , with  $\alpha \in typ(A)$  and  $\beta \in typ(B)$ , would represent an implication of types among systems that is in accordance to how the tokens of different systems are connected between each other automatically. The proposed process distinguishes four steps to solve this problem.

##### 4.1. Identification of IF Classifications

We associate G-Types as Types of classification to PC-Tokens as Tokens. Possible PC-Tokens are:

$c1 = (pressure, SFA)$ ,  $c2 = (pressure, DFA)$ ,  
 $c3 = (velocity, FA)$ ,  $c4 = (level, FA)$ ,  
 $c5 = (velocity, ExtEnt)$ ,  $c6 = (level, ExtEnt)$ ,  
 $c7 = (level, FA, ExtEnt)$ ,  $c8 = (offset, gate)$ ,  
 $c9 = (position, gate)$ .

For each complex goal  $(\Gamma_1, \Gamma_2, \Gamma_3)$  of the first system and  $(\Gamma'_1, \Gamma'_2, \Gamma'_3)$  of the remote one. We give in Table 1 the different classifications of the first system. The second system presents the same classifications replacing  $\gamma$  by  $\gamma'$ .

##### 4.2. Construction of IF Channel

It is the central aspect in the process of alignment functional ontologies. In this step, the physical context of  $\gamma_5$ , ( $c_5$ ) is not complete because the velocity must be measured and



**Table 1.** The binary relation  $\models$  of IF classifications in hydraulic subsystem

	$F_1$				$F_2$								$F_3$	
	$\gamma_1$	$\gamma_2$	$\gamma_3$	$\gamma_4$	$\gamma_1$	$\gamma_2$	$\gamma_3$	$\gamma_5$	$\gamma_6$	$\gamma_7$	$\gamma_9$	$\gamma_8$	$\gamma_9$	
$c_1$	1	1	1	1	1	1	1	0	1	1	1	0	0	
$c_2$	1	1	1	1	1	1	1	0	1	1	1	0	0	
$c_3$	0	1	0	1	0	1	0	0	1	1	1	0	0	
$c_4$	0	0	1	1	0	0	1	0	1	1	1	0	0	
$c_5$	0	0	0	0	0	0	0	1	1	1	1	0	0	
$c_6$	0	0	0	0	0	0	0	1	1	1	1	0	0	
$c_7$	0	0	0	0	0	0	0	0	1	1	1	0	0	
$c_8$	0	0	0	0	0	0	0	0	0	1	1	1	1	
$c_9$	0	0	0	0	0	0	0	0	0	0	1	0	1	

forwarded from an external (remote) physical entity. A similar problem occurs in  $\gamma_6(c_6)$  because the level cannot be computed without having first received the velocity value. The entities in  $c_5$  and  $c_6$  are not known locally (system 1). The idea is to find in remote systems the physical contexts having the same physical roles (velocity and level). We assume a partial alignment of physical contexts ( $c_5$  and  $c_6$ ) with the physical context candidates from the remote system ( $c'_j, \dots$ ). This equivalence can be formalized with a classification A, where the types are  $c_5$  and  $c_6$ , with all their possible tokens using the grey code.

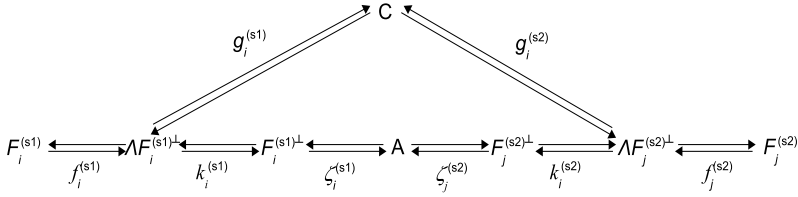
	$\models_A$	$c_5$	$c_6$
$a$		0	0
$b$		0	1
$c$		1	0
$d$		1	1

To relate the PC-Types classification with the goal classification, it is useful to introduce the flip of goal classifications where flipping amounts to interchanging rows and columns. Because we begin the process with the classification A, the flips of goal classifications is required in order to relate with initial classifications<sup>6</sup>. In order to satisfy the context correspondences, the flips of classifications  $\mathcal{F}_i^{(s1)\perp}, \mathcal{F}_j^{(s2)\perp}$  are introduced, which give rise to the respective couples of infomorphisms  $\zeta_i^{(s1)}, \zeta_j^{(s2)}$ . The purpose of conjunctive power<sup>7</sup> is to find the G-Types satisfying simultaneously the related physical contexts. The conjunctive power classifies G-Types to sets of physical contexts whenever all of their context is in the set. This gives rise to the conjunction infomorphisms:

$$k_i^{(s1)} : F_i^{(s1)\perp} \rightleftharpoons \wedge F_i^{(s1)\perp} \text{ and } k_j^{(s2)} : F_j^{(s2)\perp} \rightleftharpoons \wedge F_j^{(s2)\perp}$$

<sup>6</sup>Given an IF classification  $A : \langle tok(A), typ(A), \models_A \rangle$ , the flip of A is an IF classification:  $\langle tok(A), typ(A), \models_A^T \rangle$ , where T is the transpose of the binary relation. Its functional hierarchy is denoted:  $\mathcal{F}^\perp$

<sup>7</sup>Conjunctive power :  $\wedge A$  of an IF classification A is the classification whose tokens are the same as A, whose types are subsets of  $typ(A)$ , and given  $a \in tok(A)$  and  $\Phi \subseteq typ(A)$ ,  $a \models_{\wedge A} \Phi$  iff  $a \models_A \sigma$  for every  $\sigma \in \Phi$ . There exists a natural embedding  $\eta_A : A \rightleftharpoons \wedge A$  defined by  $\eta_A^\wedge(\alpha) = \alpha$  and  $\eta_A^\vee(a) = a$ , for each  $\alpha \in typ(A)$  and  $a \in typ(\wedge A)$



**Figure 4.** Distributed IF system

According to the example, the partial alignment is a binary relation between  $Typ(F_2^\perp)$  and the candidates  $Typ(F_1'^\perp)$  and  $Typ(F_2'^\perp)$ . In order to represent this alignment as distributed IF system in channel theory, we decompose the binary relation into functions  $\zeta_i^{\wedge(s1)}$  and  $\zeta_j^{\wedge(s2)}$  from a common domain  $Typ(A) = \alpha, \beta$ :

$$\zeta_i^{(s1)} : A \rightarrow \wedge F_i^{(s1)\perp} \text{ and } \zeta_j^{(s2)} : A \rightarrow \wedge F_j^{(s2)\perp}$$

To satisfy the property of infomorphisms, the token of  $\zeta_i^{(s1)}$  and  $\zeta_j^{(s2)}$  must be as follows:

**with  $F_2$**

$$\zeta_2^\vee(\gamma_1) = a, \zeta_2^\vee(\gamma_2) = a, \zeta_2^\vee(\gamma_3) = a, \zeta_2^\vee(\gamma_5) = d, \\ \zeta_2^\vee(\gamma_6) = d, \zeta_2^\vee(\gamma_7) = d, \zeta_2^\vee(\gamma_9) = d$$

**with  $F_1'$**

$$\zeta_1'^\vee(\gamma'_1) = a, \zeta_1'^\vee(\gamma'_2) = c, \zeta_1'^\vee(\gamma'_3) = b, \zeta_1'^\vee(\gamma'_4) = d$$

**with  $F_2'$**

$$\zeta_2'^\vee(\gamma'_1) = a, \zeta_2'^\vee(\gamma'_2) = a, \zeta_2'^\vee(\gamma'_3) = a, \zeta_2'^\vee(\gamma'_5) = d \\ \zeta_2'^\vee(\gamma'_6) = d, \zeta_2'^\vee(\gamma'_7) = d, \zeta_2'^\vee(\gamma'_9) = d.$$

This alignment allows the generation of the desired channel between  $F_i^{(s1)}$  and  $F_j^{(s2)}$ . A core classification C is build with a couple of infomorphisms :

$$g_i^{(s1)} : \wedge \mathcal{F}_i^{(s1)\perp} \rightleftarrows C \text{ and } g_j^{(s1)} : \wedge \mathcal{F}_j^{(s2)\perp} \rightleftarrows C$$

Figure 4 describes the distributed IF system.

#### 4.3. Identification of the IF Logic on the Core of IF Channel

As it is mentioned above, C is a classification, its types are elements of the disjoint union of types from  $\wedge \mathcal{F}_i^{(s1)\perp}$  and  $\wedge \mathcal{F}_j^{(s2)\perp}$  and the tokens are pairs of goal types  $(\gamma_p^{(s1)}, \gamma_q^{(s2)})$ . A token of  $\wedge \mathcal{F}_i^{(s1)\perp}$  is connected to a token of  $\wedge \mathcal{F}_j^{(s2)\perp}$  to form the pair  $(\gamma_p^{(s1)}, \gamma_q^{(s2)})$  iff  $\zeta_i^{(s1)\vee}(\gamma_p^{(s1)})$  and  $\zeta_j^{(s2)\vee}(\gamma_q^{(s2)})$  are of the same type in A.

For example, the core C will have a token  $(\gamma_5, \gamma'_4)$  connecting  $\wedge \mathcal{F}_2^{(s1)\perp}$ -token  $\gamma_5$  with  $\wedge \mathcal{F}_1'^\perp$ -token  $\gamma'_4$ , because  $\zeta_2^\vee(\gamma_5) = d$  and  $\zeta_1'^\vee(\gamma'_4) = d$  and d is a token of type

$\alpha$ . We take another example, a token  $(\gamma_5, \gamma'_2)$  connecting  $\wedge \mathcal{F}_2^{(s1)\perp}$ -token  $\gamma_5$  with  $\wedge \mathcal{F}_1^{\perp}$ -token  $\gamma'_2$ , because  $\zeta_2^\vee(\gamma_5) = d$  and  $\zeta_1^\vee(\gamma'_2) = c$ , because both of  $d$  and  $c$  are of type  $\alpha$  in **A**.

The IF classification of all connections to those types of the core that are in the image inverse of  $g_i^{(s1)} \circ f_i^{(s1)}$  and  $g_j^{(s2)} \circ f_j^{(s2)}$ , which are infomorphisms to distribute the IF logic on the core to the IF classifications  $F_2$ ,  $F'_1$  and  $F'_2$ .

In the considered example, the constraints of the IF logic on the core are written:

$$\{c'_1, c'_2, c'_3, c'_4\} \vdash \{c_5, c_6\}$$

$$\{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_9\} \vdash \{c_5, c_6\}$$

#### 4.4. Building the Distributed IF Logic

Given the logic  $\text{Log}(C)=L$  on the core  $C$ , the distributed logic  $D\text{Log}(C)$  on the sum of goal hierarchies  $F_i^{(s1)} + F_j^{(s2)}$  is the inverse image of  $\text{Log}(C)$  on this sum. In other words, the inverse image of the IF logic in  $C$  is the result of the co-product of  $F_i^{(s1)}$  and  $F_j^{(s2)}$  with the morphism  $[f_i^{(s1)} \circ g_i^{(s2)}, f_j^{(s2)} \circ g_j^{(s2)}]^{-1}$ . We obtain sequents like  $\gamma_p^{(s1)}, \gamma_q^{(s2)}$  relating goal(s) on remote systems to the local goal(s). Its IF theory highlights the following constraints :

- $\gamma'_4 \vdash \gamma_5$  for the sum  $F_2, F'_1$
- $\gamma'_6 \vdash \gamma_5$  for the sum  $F_2, F'_2$

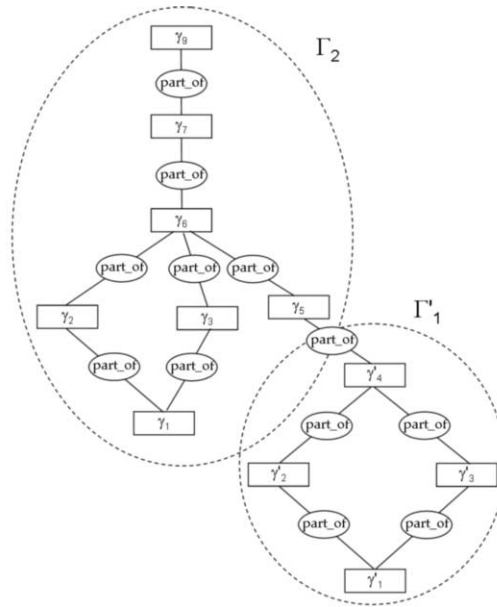
According to the initial constraint on the G-Types (i.e.,  $(\{to\_send\}, \{(velocity, X), (level, X)\}))$ , only the first sequent matches all conditions. Since the logic on the core is complete, its inverse image is complete as well. The logic is guaranteed to be sound on those tokens of the sum that are sequences of projections of a normal token of  $\mathcal{L}$ . Therefore, the goal hierarchy  $F'_1$  ( which represents the global goal  $\Gamma'_1$ ) has to be mapped to  $F_2$  (representing  $\Gamma_2$ ) in order to constitute a sound distributed service.

From this point, it is straightforward to extend goal dependencies to dependencies between higher-level goal, and finally between distributed services.

## 5. Agent Specification of Ontology Alignment

In [14], authors described an approach to ontology negotiation between agents supporting intelligent information management. They have developed a protocol that allows agents to discover ontology conflicts and then, through incremental interpretation, clarification, and explanation, establish a common basis for communicating with each other. Our aim is not to extend this work, but to clarify the idea of using intelligent agents to facilitate the automatic alignment of functional ontologies. In [15], authors have related the concepts of Service-Oriented Architectures (SOA) and the Model-Driven Architecture (MDA) to agent technologies for the solving of interoperability problems.

Agents operate with autonomy and can cooperate with other agents to perform a task, such as the achievement of the system goal(s). In other works, agents are able to locate other agents, capable of performing specific search, fusion, or filtering tasks. As



**Figure 5.** Alignment between the distributed functional ontologies

a result they (agents) are defined by their functions and missions (fusion, filtering, ..). In this work we are mainly interested in the development of deliberative agents to implement adaptive systems in open and distributed environments. Deliberative agents are usually based on a BDI model [16], which considers agents as having certain mental attitudes, Beliefs, Desires, and Intentions (BDI). An agent's beliefs correspond to information the agent has about the world (e.g., variables, facts, ...), an agent's desires intuitively correspond to the goals allocated to it and an agent's intentions represent desires (i.e., goals) that it has committed to achieving.

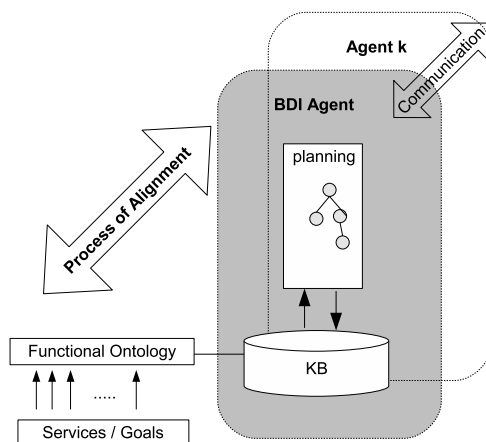
Current BDI theory and systems do not provide an architectural framework for deciding how goals interact and how an agent can decide which goals to pursue. Therefore, the purpose of this part is to specify a BDI agent model in the IF-based multi-agent modeling framework which incorporates beliefs, desires and intentions. The result is a specific BDI agent in which dependencies between beliefs, desires and intentions are made explicit within an IF-based mechanism for handling goal relationships at the architectural level. According to the proposed process of ontology alignment, agents have as:

**Beliefs B:** classifications (representing the possible services of each system), IF theory, local logic.

**Desires D:** The global goal to achieve.

**Intentions I:** available when the alignment process is complete ( it checks the goal achievement).

We proposed in [17] an algorithm describing the agent specification of the process of alignment in which every software agent represents a system and communicates by exchanging information with other distant agents. Each agent has its knowledge base (KB) containing local ontologies of the different services (goals) which can be provided



**Figure 6.** Agent specification of the alignment process

by the system (see Figure 6). Each agent is responsible for achieving services (goals), if the goal is achievable locally (in the local system), the agent proceeds it, else, the process of alignment described above is applied through information exchange between the local system and other systems. It terminates once the functional ontologies are aligned.

## 6. Conclusion

In this article, we have presented a formal mechanism for aligning distributed functional ontologies in a sound manner with major guidelines for the specification of related teleological agents. The service composition can occur either at run-time between distributed systems in order to find goal dependencies or in the design step as a modular tool where the user composes high-level goals from primitive goals. The agent specification meant to be simple enough to be used in the development of reusable and maintainable agent architectures for agent-oriented systems in engineering environments.

The notion of dependence between agents is a challenging problem [18]. Some authors have proposed a graph structure to formalize the relationships between agents [19]. The IF-based approach tackles the problem of building these dependencies from distributed logics. Ongoing works investigate the application of channel theory in industrial environments where goal structures generalize the role concept to the industrial or business framework and where the physical context is replaced by a business context.

## References

- [1] P. Giorgini, E. Nicchiarelli, J. Mylopoulos, R. Sebastiani: Reasoning with goal models. Proceedings of the international conference on conceptual modeling (2002) 167-181.

- [2] R. Fikes: Ontologies: What Are They, and Where's The Research?; Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning; Cambridge, Massachusetts; November 5-8, (1996) 652-654.
- [3] Y. Kalfoglou, M. Schorlemmer: Ontology Mapping: the state of the art. *The Knowledge Engineering Review*, 181, (2003) 1-31.
- [4] J. Barwise, J. Seligman: *Information Flow*. Cambridge tracts in Theoretical Computer Science, **44**, Cambridge University Press(1997).
- [5] N. Noy, H. Stuckenschmidt: Ontology Alignment: An annotated Bibliography, Dagstuhl Seminar Proceedings 04391, (2005).
- [6] A. Doan, A.Y. Halevy: Semantic Integration Research in the Database Community: A Brief Survey, *American Association for Artificial Intelligence aaai*(2004) 83-94.
- [7] A. Zimmermann, M. Krötzsch, J. Euzenat, P. Hitzler: Formalizing Ontology Alignment and its Operations with Category Theory. Baltimore, Maryland (USA), November (2006), 9-11.
- [8] Y. Kalfoglou, M. Schorlemmer: IF-Map: An ontology mapping method based on Information Flow theory. *Journal of Data Semantics*, S.Spaccapietra et al eds., **11**, Springer Verlag, (2003) 98-127.
- [9] Y. Kalfoglou, M. Schorlemmer: Formal Support for Representing and Automating Semantic Interoperability. In *Proceedings of the 1st European Semantic Web Symposium (ESWS'04)*, Heraklion, Crete, May (2004) 45-60.
- [10] Y. Kalfoglou, M. Schorlemmer, M. Uschold, A. Sheth, S. Staab: Semantic Interoperability and Integration. In *Dagstuhl seminar 04391 on Semantic Interoperability and Integration materials*, Dagstuhl, Germany, October (2004).
- [11] L. Chittaro, G. Guida, C. Tasso, E. Toppano: Functional and Teleological Knowledge in the Multimodeling Approach for Reasoning about Physical Systems: A case study in diagnosis *IEEE Transactions on Systems, Man, and Cybernetics*, **23**(6) (1993), 1718-1751.
- [12] M. Lind: Modeling Goals and Functions of Complex Industrial Plant. *Journal of Applied Artificial Intelligence* 8 (1994), 259-283.
- [13] R. Dapoigny, P. Barlatier, N. Mellal, E. Benoit, L. Foulloy: Inferential Knowledge Sharing with Goal Hierarchies in Distributed Engineering Systems. *Proceedings of IIAI'05, Pune (India)*(2005) 590-608.
- [14] C. Bailin, W. Truszkowski: Ontology Negotiation between Agents Supporting Intelligent Information Management. *Workshop on Ontologies in Agent Systems*(2001) 7-19.
- [15] K. Fischer, A. Berre: Model-Driven Development of Multiagent Systems in Service-Oriented Architectures. *The Eighth European Agent Systems Summer School EASSS, Annecy (France)* July 2006, 17-21.
- [16] A.S. Rao, M.P. Georgeff: BDI Agents: from Theory to Practice. In *Proceedings of the 1st International Conference on Multi-agent Systems (ICMAS'95)*(1995), 312-319.
- [17] N. Mellal, R.Dapoigny: A Multi-Agent Specification for the Goal-Ontology Mapping in Distributed Complex Systems. *International conference on Data Base and Information System DBIS'06 Lithuania* (2006) 235- 243.
- [18] C. Castelfranchi, M. Miceli, A. Cesta: Dependence relations among autonomous agents. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3*. Elsevier Science Publishers, (1992) 215-227.
- [19] J.S. Sichman, R. Conte: Multi-Agent Dependence by Dependence Graphs. *Proceedings of AAMAS'02*, (2002) 483-490.

# Principles of Model Driven Architecture in Knowledge Modeling for the Task of Study Program Evaluation

Oksana NIKIFOROVA<sup>1</sup>, Marite KIRIKOVA and Natalya PAVLOVA  
*Riga Technical University, Institute of Applied Computer Systems, Latvia*

**Abstract.** Two-hemisphere model driven (2HMD) approach assumes modeling and use of procedural and conceptual knowledge on equal and related basis according to the principles of Model Driven Architecture (MDA), which separates different aspects of system modeling. This differentiates 2HMD approach from pure procedural, pure conceptual, and object oriented approaches. The approach may be applied in the context of modeling of a particular business domain as well as in the context of modeling the knowledge about the domain. Therefore, the principles of MDA via 2HMD approach may be applied not only in the context of software development but also in the context of the study course and program development. Knowledge modeling by 2HMD approach gives an opportunity to transparently analyze and compare knowledge to be provided and knowledge actually provided by courses belonging to a particular study program, and, thus, to identify and fill gaps between desirable and actual knowledge content of the study program.

**Keywords.** Model Driven Architecture, knowledge modeling, education

## Introduction

The Model Driven Architecture (MDA) is a framework being built under supervision of the Object Modeling Group (OMG) [1]. MDA separates the system business aspects from the system implementation aspects on a specific technology platform. MDA defines the approach and tool requirements for specifying systems independently of platforms, specifying platforms, choosing particular platform for the system and transforming business domain specification into one for the chosen platform. MDA proposes a software development process in which the key notions are models and model transformation. In this process, software is built by constructing one or more models, and transforming these into other models. The common view on this process is that the input models are platform independent and the output models are platform specific, and that the platform specific models can be easily transformed into a format that is executable [2].

In this paper we apply principles of MDA in the domain of education, namely, for the task of study program evaluation. Each study program may be viewed as a “platform” specific knowledge, because it is delivered by means of particular courses,

---

<sup>1</sup> Corresponding Author: Oksana Nikiforova, Riga Technical University, Kalku 1, LV 1648, Riga, Latvia;  
E-mail: oksana.nikiforova@cs.rtu.lv

which are taught by particular teachers using particular study equipment. The study program has to be updated continuously. Therefore it has to be frequently evaluated against different academic and industrial requirements or standards. Like study programs these requirements may be viewed as platform dependent knowledge, where the “platform” reflects viewpoints and beliefs of requirements and standards developers.

The principles of MDA for the task of study program evaluation are applied using Two-hemisphere model driven (2HMD) approach that has been successfully used in the framework of MDA for domain modeling and software design [3, 4, 5]. The 2HMD approach may also be applied to the task of knowledge modeling for educational purposes [6]. This task becomes more and more important thanks to rapid knowledge growth and complexity of knowledge provision in knowledge based economy.

The goal of the paper is to show the way how the problem of study program evaluation against conceptually differently structured requirements or standards can be solved by using 2HMD approach for knowledge modeling. From the point of view of 2HMD approach each course of the study program may be considered as a knowledge provider, which fulfills particular knowledge requirements. The particular study program oriented (or “dependent” in terms of MDA) knowledge requirements are derived from the knowledge model that consists of functional and conceptual “hemispheres”, which reflect knowledge derived from original academic or industrial requirements or standards. To balance formal aspects of the description of 2HMD approach application for knowledge domain and easiness of understandability of the approach, all issues described in the paper are illustrated on the basis of SWEBOK, which can be considered as a body of core [7] knowledge in Software Engineering needed for successful career development in the area of computer science and information technology [8].

Features of knowledge models, which are suitable for curricula analysis and development, are discussed in Section 1. Main principles of Model Driven Architecture and modification of 2HMD approach for knowledge modeling and the method of its use in curricula evaluation are presented in Section 2. An example of application of the 2HMD approach for evaluation of study program “Computer Systems” is illustrated in Section 3. This section is divided into two subsections according to the steps of the proposed evaluation algorithm. Brief conclusions are presented at the end of the paper.

## **1. Modeling Knowledge for Knowledge Provision**

The need to provide knowledge timely and accurately arises in different settings, ranging from industrial needs to train employees in new technologies and ending with universities that have to become more and more flexible in terms of their curricula and course development [7]. For example, undergraduate Information Systems Model Curriculum, which was first developed in 1972, has been updated in 1982, 1997, and 2002 [9]. So the last update was made in a two or three times shorter time period than the previous ones. This tendency of more frequent changes in curricula and course contents requires special means for course and curricula maintenance in terms of knowledge modeling and provision.

In industrial setting the following classification of knowledge has been introduced and is used to analyze company’s competitive position [10, 11]:

- core knowledge;
- advanced knowledge;



- innovative knowledge.

Core knowledge is the basic scope and level of knowledge that represent a basic industry knowledge barrier to entry. Advanced knowledge enables the firm to be competitively viable. It is company's superior knowledge in certain areas. Innovative knowledge allows company to lead the entire industry.

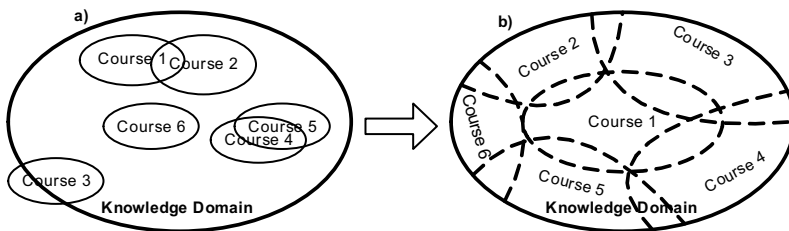
The same classification of knowledge may be applied to the contents of curricula like [12] or [13] and courses in continuously changing knowledge areas to characterize the competitiveness of a curriculum and, consequently, the competitiveness of students in the labor market. Thus, from the point of view of curriculum, the core knowledge is knowledge that guarantees students' ability to successfully enter the labor market, advanced knowledge provides background for advanced positions in the labor market, and innovative knowledge supports opportunities to obtain leading positions in the industrial labor market [7].

Taking into consideration partitioning of knowledge given above, it is obvious that knowledge model should be developed in a manner that, on the one hand, represents all three types of knowledge, and, on the other hand, gives an opportunity not only to distinguish between those types of knowledge, but also to move knowledge from one type to another when a particular piece of advanced knowledge becomes core knowledge and innovative knowledge becomes advanced knowledge [11]. This feature of knowledge model partly is achieved by module principle in curriculum development.

However, the knowledge is represented in module description as traditional sentence-based expressions that allow a limited level of transparent detail of knowledge representation. A higher level of detail may be achieved when course descriptions are added to the module description; however, it does not provide any formal means for consistency and completeness check of knowledge actually provided. Therefore educational situation may easily turn into the situation depicted in the ellipse a) of Figure 1 [6].

Moreover, informally represented knowledge is hardly comparable to particular knowledge standards [7] because of differences in terms, granularity of knowledge and variety of possible interpretations. Therefore informal descriptions do not support manageability of knowledge to be provided in a frequently changing educational environment. When knowledge is manageable an ideal study situation can be aimed at the state where all necessary knowledge is included in the courses with reasonable overlapping among courses stimulating understandability of a knowledge domain [14] as shown by the ellipse b) of Figure 1.

The research reflected in this paper uses an analogy between knowledge to be provided by a particular course and a service to be provided by a software program [15].



**Figure 1.** Knowledge to be provided

This analogy suggests a hypothesis that knowledge domain may be modeled in a way similar to methods used in requirements or software engineering, e.g., object oriented or process oriented methods. To examine the hypothesis the SWEBOK - SoftWare Engineering Body Of Knowledge [8] was chosen as body of core knowledge for students of Computer Systems study program at Riga Technical University. The purpose of SWEBOK is to provide a consensually-validated characterization of the bounds of the software engineering. It does not focus on the rapidly changing technologies, although their general principles are described in relevant knowledge areas. The emphasis on engineering practice leads SWEBOK toward a strong relationship with the normative literature.

Knowledge in SWEBOK is provided in three different ways as follows [6]:

- taxonomy of knowledge areas using part-of relationships;
- short descriptions of elements reflected by the taxonomy;
- hyperlinks between concepts used in taxonomy and descriptions.

The original way of representation of SWEBOK knowledge has the same limitations as ordinary sentence-based representations discussed above. The task to develop deeper taxonomy, based on descriptions of taxonomy elements was given to 50 master students. Analysis of their work revealed not only a considerable variety of interpretations of the descriptions in terms of part-of relationships between the concepts, but also showed that, at deeper levels of the taxonomy, duplications of the concepts are unavoidable. This suggested that pure semantic or concept oriented ontology driven approaches are not suitable for modeling software engineering knowledge.

Semantic analysis of the taxonomy elements descriptions revealed that SWEBOK knowledge indirectly reflects a particular software engineering process which can be shown by process model. However, pure process oriented approach was inconvenient because of complexity of links between the sub-processes.

Moreover, according to SWEBOK, the links themselves were related in meaningful conceptual structures for each sub-sequence of sub-processes. Pure process models do not provide means for reflection of these structures. Therefore 2HMD approach [3] was chosen for knowledge representation. The approach does not restrict the knowledge model to the core knowledge only. It is applicable for reflection of advanced and innovative knowledge as well.

Another advantage of 2HMD approach is above mentioned possibility to reflect the software engineering process behind the SWEBOK description. In other words, the 2HMD approach allows to some extent to “emancipate” the knowledge provided by the standard from the “platform” of original taxonomies imposed by viewpoints, beliefs, and traditions of requirements and standards developers. This feature of 2HMD approach is very essential in the evaluation of European study programs against curriculum guidelines based on study programs in the United States and Canada [16].

## **2. Applying of MDA Principles in Two-Hemisphere Model Driven Approach for the Task of Study Program Evaluation**

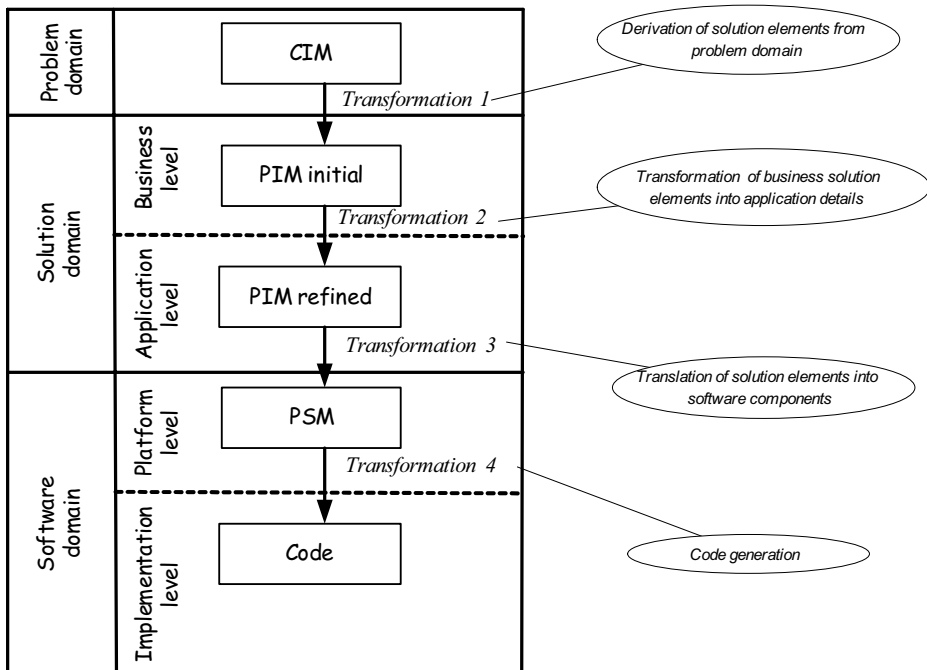
One of the modern research goals in software engineering is to find a software development process, which would provide fast and qualitative software development. Most of currently proposed methodologies and approaches try to make the development process easier and still more qualitative. For achievement of this goal the

role of explicit models becomes more and more important. Lately, the most popular approach is Model Driven Architecture (MDA) [17].

Similarly the goal of education in knowledge based economy is to provide fast and qualitative education. Therefore it is potentially promising to try to achieve it in a similar to software development way, i.e., by applying the principles of MDA.

MDA introduces an approach to system specification that separates the views on three different layers of abstraction: high level specification of what the system is expected to do (Computation Independent Model or CIM), the specification of system functionality (Platform Independent Model or PIM) and the specification of the implementation of that functionality on a specific technology platform (Platform Specific Model or PSM).

More specific system model refinement and evolution in the framework of MDA is presented in Figure 2. CIM presents specification of the system at problem domain level and can be transformed into initial elements of PIM. PIM provides formal specification of the system structure and functions that abstracts from technical details, and thus presents solution aspects of the system to be developed. Development of the solution domain model PIM initial starts at the moment when all the necessary elements are derived from problem domain description (*Transformation 1*). The PIM initial further has to be refined (*Transformation 2*) in PIM refined, which enables model transformation (*Transformation 3*) to the platform level, named Software Domain in Figure 2.



**Figure 2.** General structure of model transformation in the framework of MDA

Table 1. Software architecture vs. knowledge architecture

Key points of model presentation and transformations in terms of software architecture	Key points of model presentation and transformations in terms of knowledge architecture
<b>Problem Domain</b>	
Problem domain	Software engineering knowledge in SWEBOK
<b>Solution domain (Business level)</b>	
Solution domain	Software engineering oriented study program (curricula)
Functional model of problem domain	Process model of software engineering
Conceptual model of the problem domain	SWEBOK conceptual structure
<b>Solution domain (Application level)</b>	
Executable processes	SWEBOK processes chosen for the purposes of study programs
Object interaction	Exchange of knowledge flows in studying particular parts of SWEBOK
Objects classes	Knowledge units, which encapsulate an internal structure (attributes) and behavior (methods)
Class diagram	Software engineering knowledge architecture for further implementation in the study program
<b>Software domain (platform level)</b>	
Software architecture	Software engineering knowledge architecture
Platform specific details	Available resources and basis for study program implementation (teaching staff, infrastructure, goals, equipment, etc.)
<b>Software domain (implementation level)</b>	
Software components	Implementation of software engineering architecture in the courses of a study program, according to available platform

Similar transformations may be considered in educational domain with respect to models of knowledge to be provided by educational programs. The similarity is illustrated in Table 1, which makes a direct projection of software architecture elements into knowledge architecture elements in the context of SWEBOK.

The details in the left column of the Table 1 correspond to the 2HMD approach [3], which addresses the construction of information about problem domain by the use of two interrelated models at problem domain level, namely, the Business process model (Model of systems functioning) and the Conceptual model. The Conceptual model is used in parallel with Business process model to cross-examine software developers understanding of procedural and semantic aspects of problem domain (Figure 3). Parallel use of both models ensures high quality models, from which formalized transformation are to be started. An explicit and detailed mapping of 2HMD strategy into the framework of MDA presented in Figure 2 is shown in Figure 4.

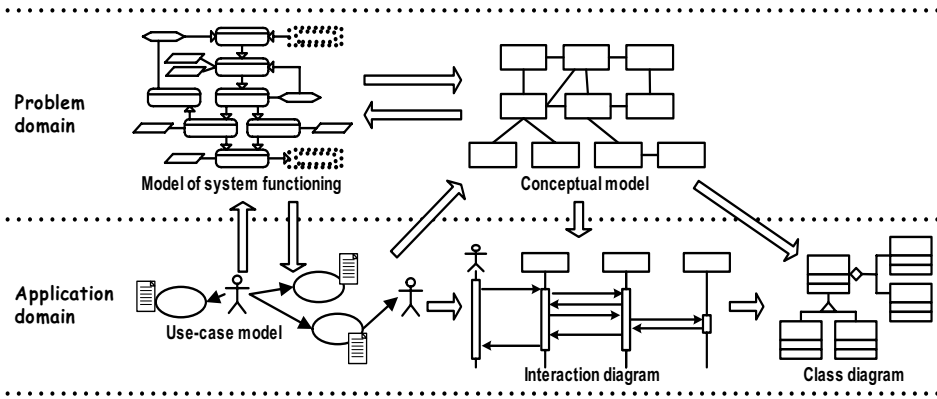


Figure 3. Original version of 2HMD approach [3]

Clear analogy between software architecture and knowledge architecture presented in Table 1 allows assuming, that several aspects of MDA and 2HMD approach can be used for knowledge modeling in the task of study program evaluation. In this paper by the term evaluation of study program we understand a comparison of existing knowledge to be provided to a particular industrial or academic requirements or standards such as, for example, SWEBOK. Our aim is to develop Solution domain Business level PIM initial (from which formalized model transformation starts in our approach) for the particular standards not for the particular knowledge domain as such. Thus, in this paper, we develop a two-hemisphere model for SWEBOK, but we do not aim at development of the model of Software Engineering discipline as such [18].

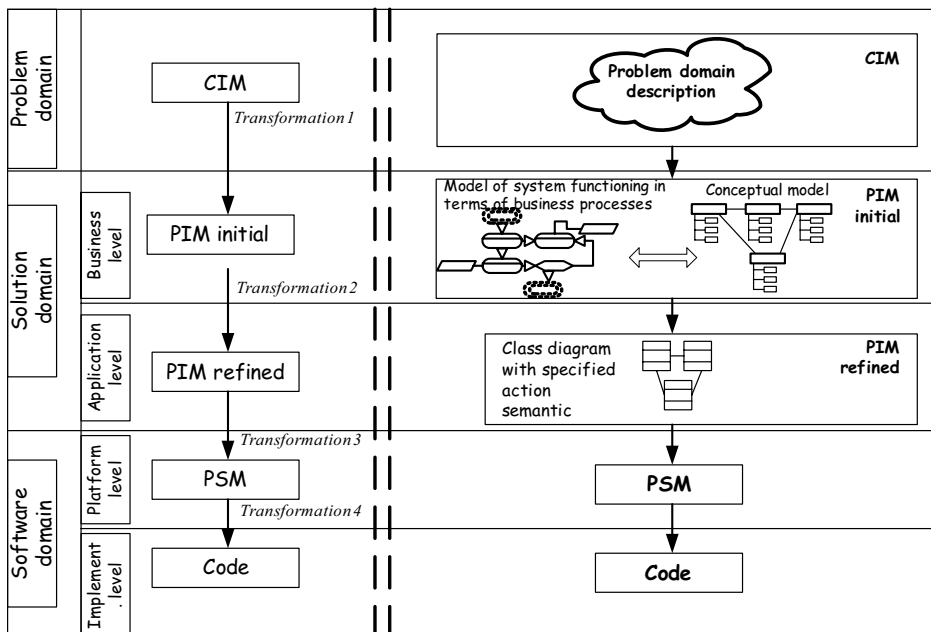


Figure 4. Mapping of 2HMD approach into framework of MDA

In the original 2HMD approach, Application domain Use-cases diagram is derived from process model and possible interactions between software architecture elements defined on the basis of Use-case model and Conceptual model (Figure 3). While software and knowledge architectures have proven similarities [15], in 2HMD approach particular differences between software development and knowledge provision domains are to be considered. Due to the difference of problem domains, the task of knowledge modeling for knowledge architecture development can be more straight-forward than software architecture development. In case of knowledge architecture all knowledge reflected in problem domain model has to be presented also at the application domain level instead of looking for processes to be automated as it is in the task of software development. Therefore 2HMD approach [3] may be modified to meet knowledge provision needs more closely.

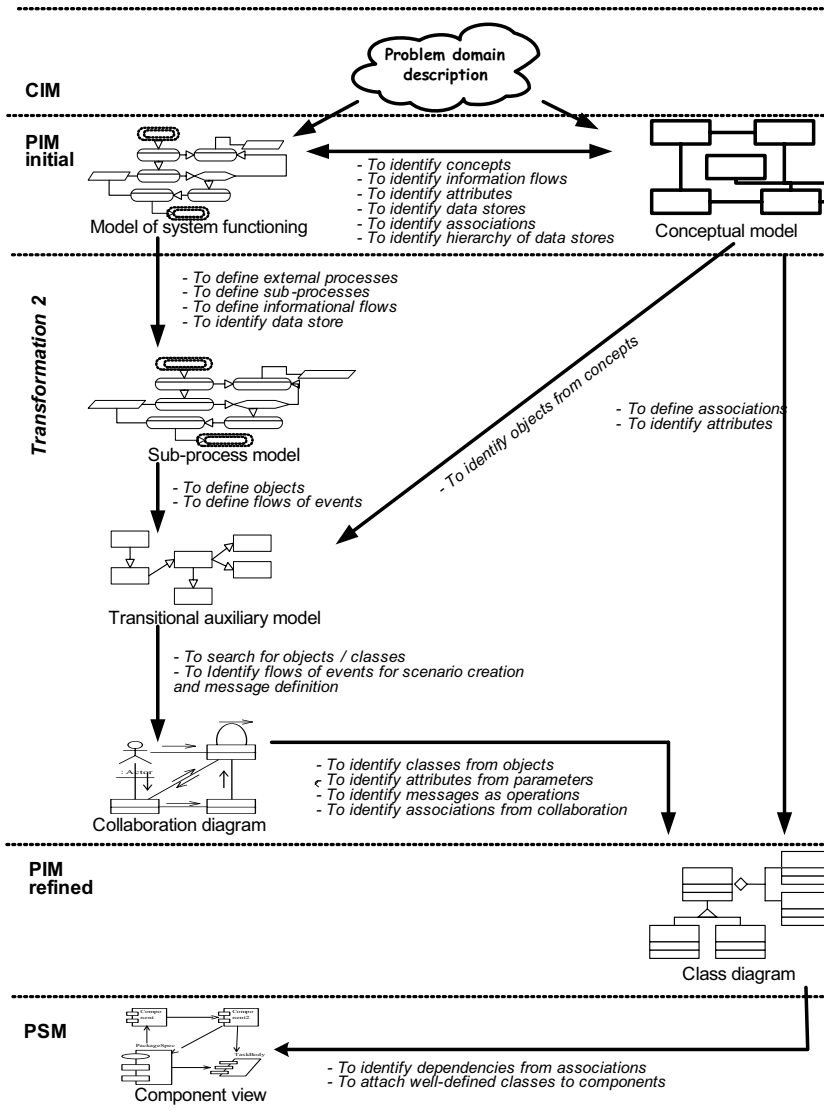
Information flows coming from one diagram into another in “two-hemisphere” (Model of system functioning and Conceptual model) representation are shown in Figure 5 [19] according to new context of knowledge architecture development and adaptation of 2HMD approach for the task of study program evaluation.

The first major change is removing use case diagram from modeling process [19]. The reason is already discussed above – as regards knowledge architecture there is no need to look for process to be automated because all the processes have the same status. This is an advantage in comparison with the original 2HMD approach where modeling of application domain begins with a use case diagram derived from business process diagram that may result in the loss of information as well as introduction of ambiguous information in the model [20].

Instead of Use-case model here the sub-models of Model of system functioning and Transitional auxiliary model are used. Those models form a formal base for further design. Transitional auxiliary model is generated from Model of system functioning using theory of graph transformation and synthesis [21], [22]. The nodes of sub-process model become arcs of transitional auxiliary model, and arcs of Model of system functioning become nodes of transitional auxiliary model [19], [20].

Secondly, collaboration diagram is added as a logical transition from Model of system functioning to present interaction of knowledge objects. Information flows from collaboration diagram to class diagram are the same as from interaction to class diagram, as shown in the Figure 4 [19].

As in the original 2HMD approach [3], information into class diagram is transferred from conceptual model and object interaction diagram. Class diagram is a final model for application domain and in the case of knowledge modeling shows knowledge classes, their structure, methods and relationships. Application of 2HMD approach for knowledge architecture development can be divided into three steps: knowledge architecture development (described in Section 2.1), evaluation of a particular study program (described in Section 2.2), and development of recommendations for study program improvement or creation. The formal description of the third step of the approach is beyond the scope of this paper.

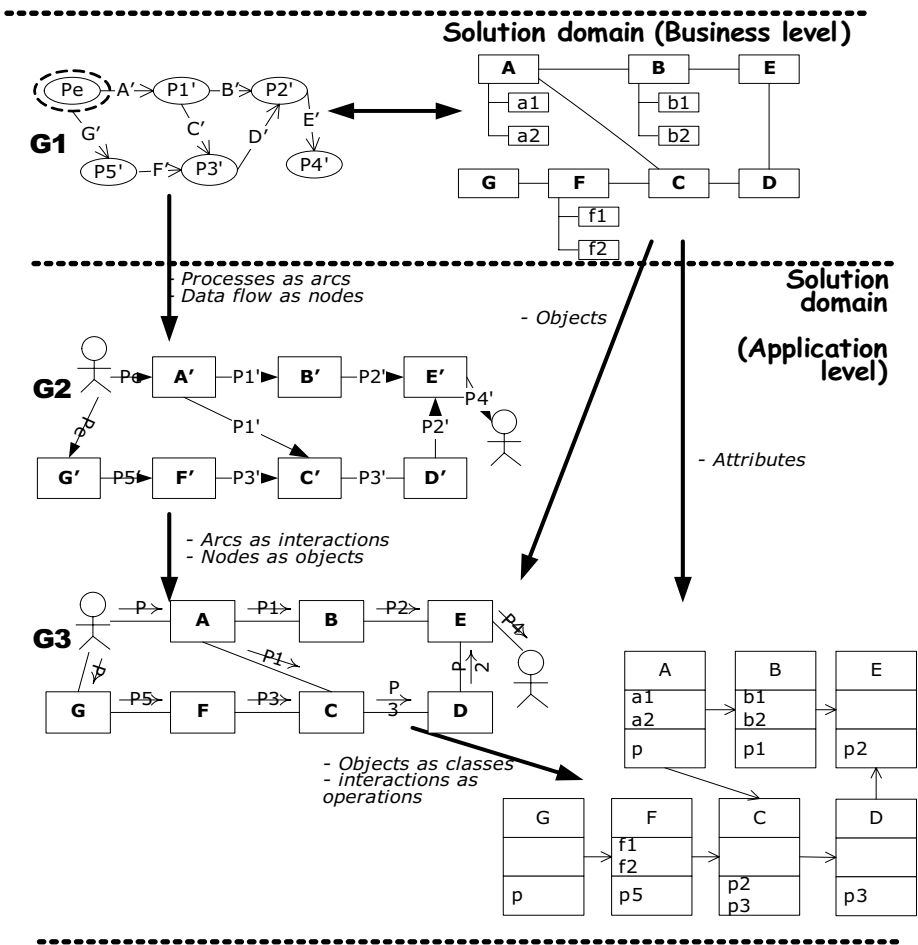


**Figure 5.** Information flows coming from problem domain level of knowledge representation into application level according to 2HMD approach

### 2.1. The First Step: Development of Knowledge Architecture

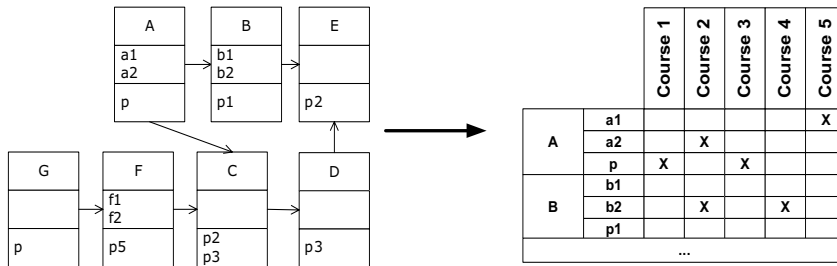
The main purpose of building knowledge architecture is to develop transparent and manageable model of knowledge area to be included in the study program to make the program complete and consistent. To achieve this, modified 2HMD approach has to be applied through two levels of the solution domain: namely business level and application level, according to transformations shown in Figure 4 and Figure 5. The formal way of building knowledge architecture at the higher level of abstraction is shown in Figure 6.

The models of 2HMD approach are represented as graphs. The first graph G1 (P', U') corresponds to business process model, where  $P' = \{Pe', P1', P2', P3', P4', P5'\}$  is the set of system business processes, and  $U' = \{A', B', C', D', E', F', G'\}$  is the set of information flows among the processes. The process  $Pe'$  is external process. The following graph G2 (U, P), which corresponds to transitional auxiliary model, consists of the same sets, however, in this graph, the set of processes comprises arcs and the set of information flows – nodes of the graph. This transformation was performed to simplify transition from business process model (graph G1) to collaboration diagram (graph G3). Graph G3 (P, U) corresponds to collaboration diagram, as discussed above. There is a set of nodes  $U = \{A, B, C, D, E, F, G\}$ , which represent objects, received from the set  $U'$ , and set of arcs  $P = \{Pe, P1, P2, P3, P4, P5\}$  which represent interactions between objects, received from the set  $P'$ . During transition from G1 to G3 through G2 each process and each data flow was transferred into collaboration diagram [20].



**Figure 6.** Formal transformation of knowledge from the solutions domain business level into the application level





**Figure 7.** Definition of knowledge content in the form of table for expert evaluation (implementation level)

The next state of transition is class diagram. Here all the elements of the set P should be operations of class, and all elements of the set U should be classes or instances of classes. In the class diagram the set of attributes is presented. Initially the attributes (set  $a=\{a1,a2\}$ ) are shown in the conceptual model, where they are shown as properties of elements of the set U. Attributes are the same and associations are defined based on the graph G3, taking into consideration that if there is an arc between A and B, then there is an association between A and B in the class diagram [20].

## 2.2. The Second Step: Evaluation of a Study Program

The class diagram developed during the first step of modified 2HMD approach can be applied to evaluate a particular study program, which is meant for the given field of knowledge. For this purpose the problem domain level knowledge representation of 2HMD approach is sufficient. The solution of the task is illustrated in Figure 7.

The evaluation is based on the table shown on the right side of Figure 7. The table contains class names that correspond to titles of knowledge items, their structure (a1, a2, ...) and processes performed with the items (p1, p2, ...) derived from class diagram, shown on the left side of Figure 7.

The table in the form of questionnaire is given for evaluation to experts. An expert puts mark "X" in the cell of the table if she/he gives a positive answer to the question "Does the Course i ... cover the required element of knowledge architecture?". Thus the overlapping of courses and missing items of knowledge are detected [6].

## 3. Application of Two-Hemisphere Model Driven Architecture for the Evaluation of Study Program "Computer Systems"

A simplified example of 2HMD approach in SWEBOK knowledge modeling and analysis of a study program is shown in Figure 6. In the example Conceptual model is presented as data structures. Such solution was imposed by business process modeling tool GRADE [23], which has built-in features for supporting relationships between links in the process model and corresponding data structures. This does not allow to apply 2HMD in full extent, but is sufficient for achieving applicable results.

Only a small part of SWEBOK "Software Requirements" knowledge is shown in Problem domain models (upper part of Figure 8).

The table of program evaluation resulting from that part of knowledge is shown in the lower part of Figure 8 [6]. The complete table of knowledge architecture elements

was given for evaluation for students studying Computer Systems and for teachers teaching particular courses in this program. The table shown in Figure 8 presents only student evaluations. The results of evaluation of study program Computer Systems according to the full model of software engineering knowledge is presented in [24].

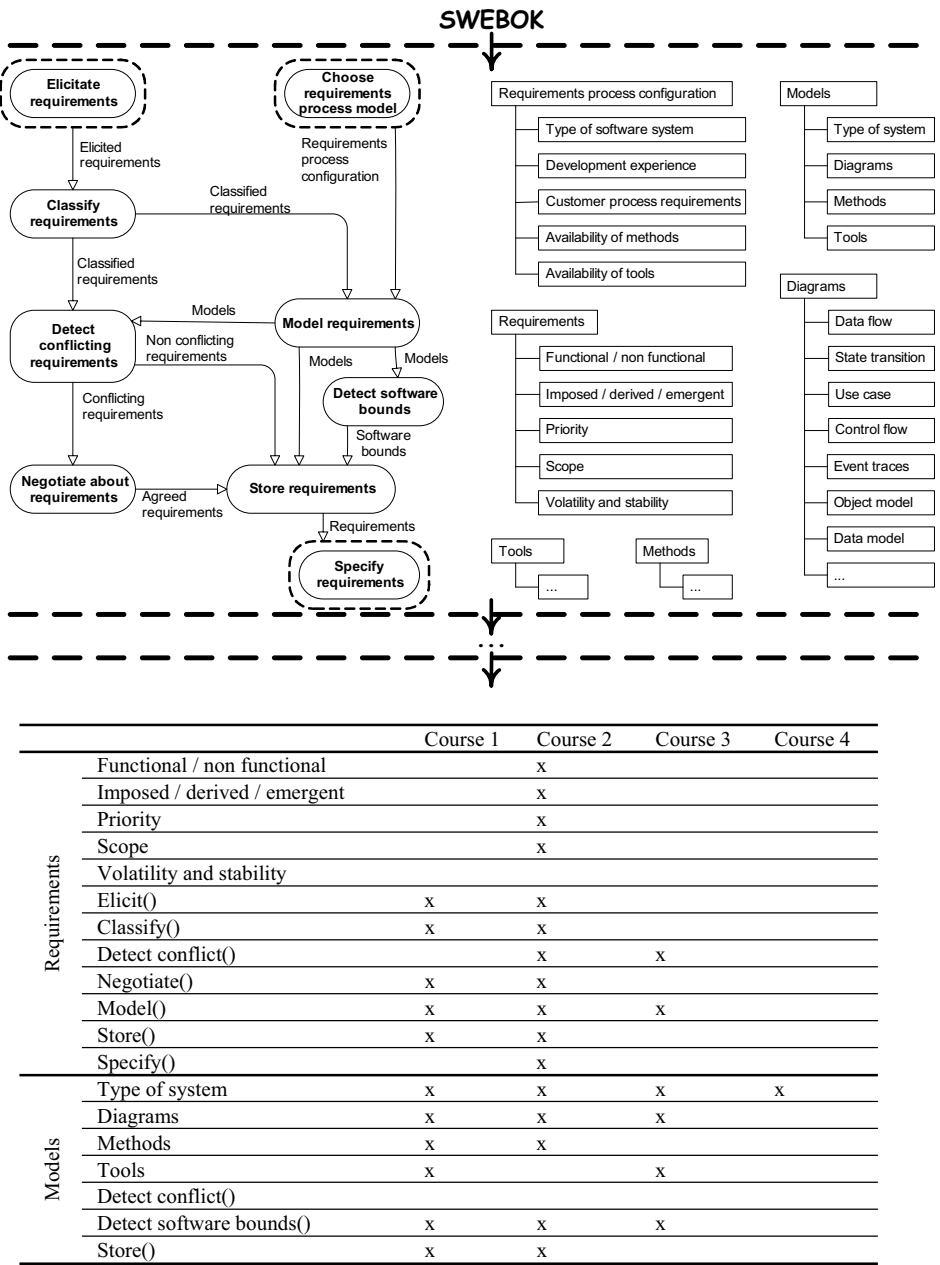


Figure 8. Mapping of the part of Software Requirements analysis on study program of Computer Systems

#### 4. Conclusion

Theoretical and practical experiments with 2HMD approach showed (1) that analogy between knowledge to be provided by a particular course and a service to be provided by a software program is useful for developing knowledge modeling approaches and (2) that the knowledge domain may be modeled by methods based on MDA principles. Knowledge modeling and architecture development at the high level of abstraction can be performed by using formal methodologies applied in software development, because in both areas the term of architecture is defined in a similar way – it is a set of components with their own structure, relationships to other components and interfaces for cooperation.

The 2HMD approach was selected in order to try to eliminate some disadvantages in presentation of knowledge in pure object oriented or pure process oriented methods. By using 2HMD approach knowledge architecture for SWEBOK was developed and applied for evaluation of study program “Computer Systems” to detect whether all elements of knowledge architecture are covered by the courses and whether there is not an unnecessary overlapping of the courses of the program. Two-hemisphere model based representation of knowledge is manageable, transparent and can be easily modified for generating new evaluation tables. This gives an opportunity to discuss program contents whenever any changes in courses are needed and, due to transparency of models, achieve consensus among teachers regarding course contents.

The research results presented in this paper may be useful for scientists researching in areas of knowledge modeling, provision and architecture development, as well as for wide spectrum of educational specialists. Currently available systems development tools do not support implementation of MDA principles reflected by 2HMD approach fully, therefore further research is needed for developing new tools or several modifications of the approach – each for a particular well-known system development tool. Manual use of the approach is not intended.

#### Acknowledgements

The research reflected in the paper is supported by the research grant No. ZP/2005-02 of Riga Technical University “Application of Two-hemisphere approach for development of flexible architecture for Software Engineering Body of Knowledge” and by the European Social Fund within the National Programme “Support for the carrying out doctoral study program's and post-doctoral researches”.

#### References

- [1] MDA Guide Version 1.0.1. [cited 2006 October 1]. Available at: <http://www.omg.org/docs/omg/03-05-01.pdf>.
- [2] Kleppe A. MCC: a model transformation environment. In: *Proceedings of the ECMDA-FA'2006*, A. Rensink and J. Warner (Eds.), LNCS 4066, Springer-Verlag, 2006, 173-187.
- [3] Nikiforova O., Kirikova M. Two-hemisphere driven approach: engineering based software development. In: *Advanced Information Systems Engineering*, Proceedings of the 16th International Conference CAiSE 2004, Riga, Latvia, June 2004, A. Persson and J. Stirna (Eds.), Springer Verlag, Berlin Heidelberg, 2004, 219-233.
- [4] Nikiforova O., Kirikova M. Enabling problem domain knowledge transformation during object-oriented software development. In: *Constructing the Infrastructure for the Knowledge Economy: Methods and*

- Tools, Theory and Practice*, H.Linger, J. Fisher, W. Wojtkowski, W.G. Wojtkowski, J.Zupancic, K.Vigo, J.Arnolds (Eds.), Kluwer Academic/Plenum Publishers, 2004, 481-495.
- [5] Nikiforova O., Kirikova M., Wojtkowski W. Role of models in knowledge transfer during OO software development. In: *Proceedings of the 15th European-Japanese Conference on Information Modelling and Knowledge Bases*, Tallinn, Estonia, May 16-20, 2005, Y.Kiyoki, H.Kangassalo, H.Jaakkola and J.Henno (Eds.), 305-320.
  - [6] Nikiforova O., Kirikova M., Pavlova N. Two-hemisphere driven approach: application for knowledge modeling. In: *Proceedings of the Seventh IEEE International Baltic Conference on DB and IS (BalticDB&IS'2006)*, O. Vasilecas, J. Eder, A. Caplinskas (Eds.), IEEE, 2006, 244-250.
  - [7] Grundspenkis J., Kirikova M., Vinogradova V. Meeting educational challenges by knowledge management. In: *Proceedings of the College Teaching and Learning Conference*, Lake Buena Vista, Florida, January 5-9, 2004, article number 392.
  - [8] Guide to the Software Engineering Body of Knowledge, IEEE-2004 version. [cited 2006 February 13]. Available at: <http://www.swebok.org>.
  - [9] Gorgone J., Kanabar V. Masters in Information Systems: a Web-centric model curriculum. *Information Science*, June 2002, 553-563.
  - [10] Zack M. H. Developing a knowledge strategy. *California Management Review*, Vol. 41, No 3, Spring 1999, 125-145.
  - [11] Tiwana A. *The Knowledge Management Toolkit: Orchestrating IT, Strategy, and Knowledge Platforms*. 2nd ed., Prentice Hall, 2002.
  - [12] Roberts E., Engel G., Chang C., Cross J.H., Shackelford R., Sloan R.H., Carver D., Eckhouse R., King W., Lau F., Srimani P., Austing R., Cover C. F., Davies G., McGettrick A., Schneider G. M., Wolz U. *Computing Curricula 2001: Computer Science*. Los Angeles, New York: IEEE Computer Society, Association for Computing Machinery, 2001.
  - [13] Gorgone J. T., Davis G. B., Valacich J. S., Topi H., Feinstein D. L., Longenecker (Jr.) H. E. *IS 2002: Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems*. Atlanta: AIS, 2002.
  - [14] Anderson J. R. *Cognitive Psychology and its Implications*. Freeman and Company, 1995.
  - [15] Tyugu E. Describing knowledge architectures. In: *Proceedings of the 15th European-Japanese Conference on Information Modelling and Knowledge Bases*, Tallinn, Estonia, May 16-20, 2005, Y.Kiyoki, H.Kangassalo, H.Jaakkola and J.Henno (Eds.), 356-361.
  - [16] Gogone J.T., Gray P., Stohr E. A., Valacich J. S., Wigand R. T. MSIS 2006: model curriculum and guidelines for graduate degree programs in Information Systems. *Communications of AIS*, Volume 17, Article 1, January 2006.
  - [17] Sendall Sh., Kozaczynski W. Model transformation: the heart and soul of model-driven development. *IEEE Software*, Vol. 20, No. 5, 2003, 42-45.
  - [18] Rosener V., Avriilioni D. Elements for the definition of a model of Software Engineering. In: *Proceedings of the 2006 International Workshop on Global Integrated Model Management*, Shanghai, China, May 22, ACM Press, 2006, 29-34.
  - [19] Pavlova N., Nikiforova O. Formalization of two-hemisphere model driven approach in the framework of MDA. In: *Proceedings of the 9th Conference on Information Systems Implementation and Modelling*, Czech Republic, Prerov, 2006, 105-112.
  - [20] Pavlova N. Several outlines of graph theory in the framework of MDA, In: *Proceedings of the 15th International Conference On Information Systems Development*, Hungary, Budapest, 2006 (to appear).
  - [21] Steen L. (Ed.). *For All Practical Purposes: Introduction to Contemporary Mathematics*. 3 ed. W. H. Freeman and Company, New York, 1994.
  - [22] Grundspenkis J. Causal domain model driven knowledge acquisition for expert diagnosis system development. In: *Lecture Notes of the Nordic-Baltic Summer School on Applications of AI to Production Engineering*, Kaunas, Lithuania, 9-14 June, 1997, K. Wang and H. Pranevicius (Eds.), Kaunas University of Technology Press, 1997.
  - [23] GRADE Tools. Web site [cited 2006 February 13]. Available at: <http://www.gradetools.com>.
  - [24] Nikiforova O., Kirikova M., Sukovskis U. Two hemisphere model driven architecture for knowledge map development in the task of study program analysis. In: *Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems*, the 5th series, Vol. 26, Riga, Latvia, RTU, 2006, 112-123 (in Latvian).

## Author Index

Adaikkalavan, R.	3	Kupfer, A.	89
Anohina, A.	263	Loudcher Rabaséda, S.	133
Balandyte, M.	189	Lupeikiene, A.	203
Barna, P.	219	Ma, H.	103
Ben Messaoud, R.	133	Mathiak, B.	89
Bossung, S.	40	Mellal, N.	277
Boulicaut, J.-F.	117	Mitasiunaite, I.	117
Boussaid, O.	133	Nemuraite, L.	147, 189
Brundzaite, R.	72	Neumann, K.	89
Caplinskas, A.	vii	Nikiforova, O.	291
Chakravarthy, S.	3	Paradauskas, B.	147
Conrad, S.	161	Pavlova, N.	291
Dapoigny, R.	277	Pichler, H.	57
Dasari, R.	3	Roberts, B.	233
Eckstein, S.	89	Samia, M.	161
Eder, J.	vii, 57	Schewe, K.-D.	103
Foulloy, L.	277	Schmidt, J.W.	40
Frasincar, F.	219	Sehring, H.-W.	40
Grundspenkis, J.	263	Skusa, M.	40
Gudas, S.	72	Smaizys, A.	175
Guizzardi, G.	18	Störmann, B.	89
Haav, H.-M.	249	Svirskas, A.	233
Houben, G.-J.	219	Tammet, T.	249
Hupe, P.	40	Tonkunaite, J.	147
Ignatiadis, I.	233	Varakala, S.	3
Kääramees, M.	249	Vasilecas, O.	vii, 175
Kadarpik, V.	249	Vielgut, S.	57
Kirchberg, M.	103	Wilson, M.	233
Kirikova, M.	291		

This page intentionally left blank